COMPUTE

Microprocessor Users Group Newsletter

Volume 4 Number 2 March/April 1978 Price $1.50

TABLE OF CONTENTS

1 Complex Peripherals Course
1 National's 2900 30% to 50% Faster
2 National Introduces the INS8900
2 Microcomputer Spares
3 An IMP-16 Microcomputer System
Part 4: Interfacing
9 Software Updates
9 UDS & Pace Update
10 Multi-Microprocessor Software
11 Downloader
12 SC/MP Introkit and Keyboard Kit Expansion
14 SC/MP Motherboard Kit
14 Low Cost Control for SC/MP Microprocessor
16 1977 Compute Index
18 Editorial
18 Special Offer
18 AMD Alternate Source Agreement
18 New Library Program SL0054A P8012ASM
19 Bit Bucket
21 Firmware ROM Bootstrap
22 SC/MP Disassembler
23 Call for Papers
23 Upcoming Conferences and Conventions for Compute
24 Back Issues
24 Simplify CRT Terminal Design with the DP8350
32 Hard Software
32 Schedule of Courses

COMPLEX PERIPHERALS COURSE

National has developed a Complex Peripherals course for the experienced microprocessor user (8080, Z80, 6800, SC/MP, etc.). This course provides design information and application ideas for National's 8080A microprocessor family, including the Floppy Disk Controller, Communications Components and other complex peripheral control components. An introduction to Floppy Disk mechanisms, serial codes, and line protocols precedes the hardware discussions.

Tuition for the three day course is $395. If you want to enroll, check the schedule on page 32, then send your training enrollment form along with a check or money order to the training center where you intend to take the course.

Or if you would like more information on the course call Al Jefferis at (408) 737-6453.

NATIONAL'S 2900 30% to 50% faster

Using a new advanced bipolar LSI process, National has developed a family of 2900-type four-bit-slice microprocessor components which are 30 to 50 percent faster than any similar designs now on the market.

Designated the IDM2900 family, the new devices, 16 in all, use a process that combines low power Schottky peripheral circuitry with proprietary high speed TRI-STATE emitter coupled logic circuitry for interface. National calls this new process "SCL."

What this means is that systems designers no longer have to make a choice between either high speed or low power. Previously bit-slice system designers could use high speed ECL devices and pay the cost of high power consumption or go with low-power Schottky devices, such as previous 2900 family designs and pay the penalty of a much lower system speed.

With National's 2900 family, systems designers can have both ECL-type speeds and LS bipolar power consumption and interface capability. With the IDM2901A four-bit-slice MPU, for example, the basic microcycle time is in the 60 to 70 nanosecond range, about one half to two thirds that of comparable 2901 designs from other companies. Moreover, execution time for a typical operation, such as an add and shift is in the 60 to 120 ns range. Power consumption for the single plus 5 volt device, however, is equal to that for standard low-power Schottky designs, about 800 to 900 milliwatts. Speed-power product on the IDM2901A from National is only 6 picojoules, which is the lowest in the industry for a four bit bipolar microprocessor.

National is second sourcing most of the major components in the industry standard 2900 family, including:

- the 2901A, a four bit bipolar microprocessor slice with a 16 word by 4 bit two-port RAM, a high speed, eight function ALU, and the associated shifting, decoding and multiplexing circuitry;
- the 2902, a high speed look ahead carry generator which accepts up to four pairs of carry propagate and carry generate signals and a carry input;

(continued overleaf)

Users Group Director .... Georgia Marszalek
Applications Consultant .... Keith Winter
Editor .... David Graves
Associate Editor .... Elizabeth Simmer
Sponsored by National Semiconductor Corporation, 2900 Semiconductor Drive, Santa Clara, California 95051

- the 2909A/2911A, four bit wide address controllers intended for sequencing through a series of microinstructions contained in read only memories or programmable ROMS;
- the 29702/03, inverting 64 bit random access memories for use in scratch pad and high speed buffer memory applications; and,
- the 29750/51 and 29760/61, field programmable 32 word by 8 bit and 256 word by 4 bit PROMS, respectively, for microprogram storage or random logic function generation.

All of the standard parts are pin and functionally compatible with devices currently on the market.

In addition to the basic process improvements National is adding a number of new components to the standard 2900 family. These include:

- the 29803, a 16 way branch controller;
- the 29811, a next address controller;
- the 29901, an octal Tri-State register;
- the 29902, a priority encoder;
- the 29903, a 16 word by 4 bit clocked RAM;
- the 29908, a quad-gated flip-flop.

Available now in sample quantities are the IDM2901A, IDM2902, IDM29702/03, IDM29750/51, IDM29760/61, IDM29803, IDM29811, IDM29902, and IDM29903, IDM29908, IDM2909A/11A and IDM29901.

National Introduces the INS8900

Using a newly developed n-channel metal oxide semiconductor process, National has gone into production on a sophisticated, high throughput 16-bit microprocessor that outperforms most present 8-bit CPU designs.

Designated, the INS8900, the 40 pin device is the newest and highest-speed member of National's PACE family of microprocessors, sporting an interrupt structure, addressing modes and logical capabilities traditionally associated with minicomputers.

Instruction execution times for most of the commonly used routines on the INS8900 are equivalent to those on advanced 8-bit designs like the 8085 and 10 to 30 percent faster than on present generation designs, such as the 2 microsecond 8080.

The INS8900 is intended for use in applications where the convenience and efficiency of 16-bit word length is desired, while maintaining the low cost inherent in one-chip fixed-instruction microprocessors. The 8900 is a true 16-bit central processing unit: it makes use of 16-bit instruction words and 16-bit data words and features a powerful, efficient and flexible set of 45 instruction types. All instructions use a single word, 16-bit format, thus reducing memory access and program storage requirements. A unique feature of the 8900 is the ability to operate on both 8- and 16-bit data words. This extends the inherent efficiency and power of a 16-bit processor to 8-bit applications.

microcomputer spares

Here is an updated list of spare parts available from the Microcomputer Service Center.

For information on ordering, pricing and delivery of spare parts, call the service center at: (408)737-6270 or 737-6279.

ASSEMBLY NO.

CARDS	NOMENCLATURE	FACTORY PRICE
980301953-001	IMP/IPC Standard Programmer Panel Card	$ 400.
980302085-001	IMP/IPC Card Reader/TTY I-Face Card w/o Firmware	$ 375.
980302085-002	IMP/IPC Card Reader/TTY/CRT I-Face Card w/o Firmware	$ 375.
980302086-001	IMP-16P Programmer I-Face Card	$ 485.
980302573-001	IPC Development CPU Card	$ 612.
980302708-001	IMP Card Reader/TTY I-Face Card w/Firmware	$ 450.
980303120-001	IPC Programmer Panel I-Face Card	$ 600.
980303122-001	IPC Card Reader/TTY I-Face Card w/Firmware	$ 550.
980303896-001	IMP/IPC Floppy Disc I-Face Card w/o Firmware (use with Shugart Model 900)	$ 400
980304640-001	IMP Floppy Disc I-Face Card w/Firmware (use with Shugart Model 900)	$ 600.
980304640-002	IPC Floppy Disc I-Face Card w/Firmware (use with Shugart Model 900)	$ 600
980304889-001	IMP/IPC Programmer Panel Card	$ 450.
980305077-001	IMP/IPC Programmer Panel I-Face Card	$ 400.
980305442-001	IMP Floppy Disc I-Face Card w/Firmware (use with Shugart Model 800)	$ 395.
980305442-002	IPC Floppy Disc I-Face Card w/Firmware (use with Shugart Model 800)	$ 395.
HARDWARE		
980304559-001	Disc Drive, 110V, 60HZ Position #1, (Shugart Model 800)	$1375.
980304559-002	Disc Drive, 110V, 60HZ Position #2 (Shugart Model 800)	$1375.
980304559-003	Disc Drive, 220V, 50HZ, Position #1 (Shugart Model 800)	$1375.
980304559-004	Disc Drive, 220V, 50HZ Position #2 (Shugart Model 800)	$1375.
400102205-001	Power Supply, Standard, +5V@18A, –12V@3A (Zentec)	$ 550.
400102205-002	Power Supply, Heavy Duty +5V@18A, –12V@3A (Zentec)	$ 710.
FIRMWARE		
935301029-001	PACE LCDS Firmware Kit 2MM5242N ROMS	$ 50.
935300994-001	SC/MP LCDS Keyboard Firmware Kit 2 ROMS (SCTDBG)	$ 50.
935300995-001	SC/MP LCDS TTY Firmware Kit 2 ROMS (SCPNB)	$ 50.
935305556-001	PACE DOS (Enhanced UDS) Firmware Kit 8 MM5204Q PROMS	$ 160.
SOFTWARE		
205104514-001	Diskette, Shugart #SA100, IBM #2305830, DYSAN #3740S, MAXELL #FD3200S. For use with Shugart Model 800 or 900.	$ 25.

An IMP-16 Micro-Computer System part4: Interfacing

by Hal Chamberlin
29 Mead St.
Manchester, NH 03104

This article continues the formal article series about the PUNIBUS implementation of the IMP-16. Although the system was designed around the IMP-16 a couple of years ago when it was King, many other 16 bit microprocessors are adaptable to it. In particular the various versions of the PACE can be connected fairly easily. The ample bus bandwidth allows future processors with up to 5 times the throughput of the IMP to be used although priorities should probably be shuffled to avoid CPU hogging when such a fast processor becomes available.

In this installment interfacing techniques for memory, programmed input/output, and direct memory access devices will be discussed. To illustrate the discussion, the circuitry and operation of an 8K word memory, keyboard interface with full interrupt will be described. These are some basic elements in any system and supporting software for each is being submitted to COMPUTE as well as Fred Holmes's user's group.

an 8k Word Memory Interface

Interfacing dynamic memory to this system is so easy that there is really no reason to consider any other kind. This and the fact that the price of the 22 pin variety of 4K dynamic RAM is so low (around $4 for "full spec" parts or $2.70 in moderate quantities for a relaxed spec version [5280 SM00233] which requires more frequent refreshing) makes memory on this system more cost effective than on any other. An 8K word board can be built for less than $150 (assuming use of the SM00233 suffix) which means that a system could be fully stuffed (64K words) for less than $1200.

Before getting into the circuitry of the memory board, let's take a look at the general characteristics of the memory IC's. Unlike earlier dynamic RAM designs, the 22 pin 4K dynamic RAM was designed to be easy to use and to minimize overhead circuitry consistent with semiconductor process limitations and high performance. Power supply requirements are +12 volts, +5 volts, and -5 volts. The majority of the memory chip's circuitry is powered from the +12 volt supply. Current drain from this supply is between 20 and 30 MA when the chip is continuously cycled. The data output tri-state buffer is powered from the +5 volt supply and draws less than 1 MA when driving a standard TTL load. The -5 volt supply biases the memory chip substrate and only draws a leakage current of a few microamps.

The 22 pin RAM has one clock input which requires a nominal 12 volt swing between ground and +12 volts. All other inputs are TTL compatible with a high threshold of less than 3 volts. Unlike most MOS devices whose inputs exhibit a purely capacitive load on the signal source, some manufacturer's 4K RAM's draw input current under some conditions. As a result, address and chip select drivers may have to be more powerful than capacitive load and signal risetime might dictate. The RAM manufacturers call the clock input CHIP ENABLE and then call the memory selection input CHIP SELECT. The 12 address lines and the chip select input are strobed by the leading edge of the clock and *latched* on the memory chip until the next clock. Thus address and chip select need not be valid throughout the memory cycle.

The 4K RAM has only one output, DATA OUT. This output is tri-state and is capable of driving two standard TTL loads. The output buffer is enabled only while the clock is high and only if CHIP SELECT was latched in the active state. The data appearing on the data out line is the complement of the data written into the RAM.

The timing diagram of a typical 4K dynamic RAM is shown in figure 1. The memory cycle starts with CHIP ENABLE switching to a high state. The 12 address lines and the chip select line must have settled to their valid states by this time. Also, they must be held for 50 to 150 NS beyond the leading edge of CHIP ENABLE.

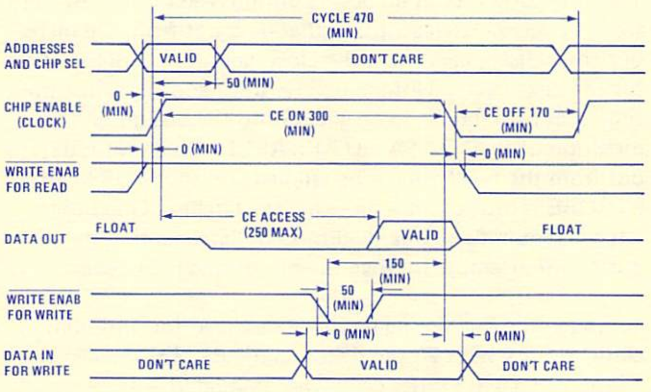


Figure 1. Typical 4K RAM Timing (MM5280-5)

Assuming a read cycle, the output is enabled and driven low shortly after CHIP ENABLE rises. After the access time elapses, the output either remains low or goes high. When CHIP ENABLE again goes low, the output is immediately disabled. If the chip select input was latched high (inactive), the output remains disabled throughout the cycle. Execution of a read cycle refreshes all 64 cells on the addressed row (address bits A0-A5) regardless of whether the chip was selected or not.

There is a minimum clock-low time requirement before the next cycle starts. During the first approximately 100 NS after CHIP ENABLE goes low, the dynamic nodes in the chip are being precharged. After this precharge period, the chip sits idle and draws little or no power.

The first part of a write cycle is identical to a read cycle. After CHIP ENABLE has gone high and the address and chip select signals have been latched, valid data is presented to the chip at the data-in pin. Coincident with valid data in, or later, the write enable pin is forced low to cause the writing operation. Timing constraints on write enable are a minimum pulse width and a minimum time interval between the leading edge of WRITE ENABLE and the trailing edge of CHIP ENABLE. Also, the input data needs to be held valid up to 40NS beyond the trailing edge of CHIP ENABLE. Writing is suppressed in chips which had latched CHIP SELECT high (inactive) but still received the write enable pulse. They will however be refreshed. The data out pin is enabled during the entire active portion of the write cycle. If the write enable pulse is delayed until the access time has elapsed, the existing data in the addressed cell will appear thus making a read-modify-write cycle possible.

The bus timing signals generated on the CPU board are exactly right for driving the memory chips just described. For example the bus signal, BUS CE, merely needs to be amplified to a 12 volt swing and applied directly to the chip enable pin of the RAM's to take care of their clocking requirements. The data bus contains address information at exactly the right time with respect to BUS CE so that it can simply be buffered and applied directly to the address pins of the RAM's. If the bus is going through a write cycle as directed by the CPU or DMA device, the data to be written and BUS WE occur at the proper time to allow direct connection to the memory chips also. Refresh, of course, is taken care of by the bus controller and thus is of no concern on a memory board.

The only difficulty at all occurs during read cycles. Recall from the previous description that data out from the memory chips disappears when the clock goes low in preparation for the next cycle. Unfortunately, this happens at just the time that read data is to be gated onto the bus under the direction of BUS DATA OUT ENABLE. As a result, data out from the RAM's must be latched and thus BUS MDR STROBE is provided for clocking the latches. One other latch is required; a "board addressed" latch so that only one board attempts to drive the bus during read cycles.

Although the bus timing has been designed for optimum compatibility with 22 pin 4K dynamic RAM's, the newer 18 pin and 16 pin types can be accomodated with little trouble. To use the TI version (TMS4050) of the 18 pin 4K RAM it will be necessary to gate the chip enable clock to select a single row of RAM's on a memory board since there is no chip select pin. Furthermore refresh cycles will have to be detected so that all RAM's on all boards will be clocked during refresh cycles. This may be accomplished with a 3-input NAND gate tied to the BUS GRANT lines.

National's version of the 18-pin RAM is more straightforward. In effect, the signals that would normally drive separate chip select and write enable pins on 22 pin RAM's are negative ORed together and applied to the Tri-Share port. One problem with both types of 18 pin RAM is that the distinction between a read and write cycle is needed earlier in the cycle than when BUS WRITE ENABLE is pulsed in order to disable the output buffer on the common data I/O line. To solve the problem either the output buffer on the RAM must be overridden by a powerful data in driver or the memory board must sample the BUS WRITE CYCLE REQUEST line and anticipate the action of the bus controller.

The increasingly popular but still expensive 16 pin RAM's will require a couple of single-shots, a two-input multiplexor, and 6 bits of latches to be added to the board. Costwise these added circuits just about make up for the clock driver that will not be needed.

Figure 2 shows a timing diagram when driven by the bus controller described in part 2 using the standard timing ROM's and Figure 3 shows the control circuitry for an 8K word memory board. Standard TTL parts are used with optional pull-up resistors for the older high threshold RAM's.

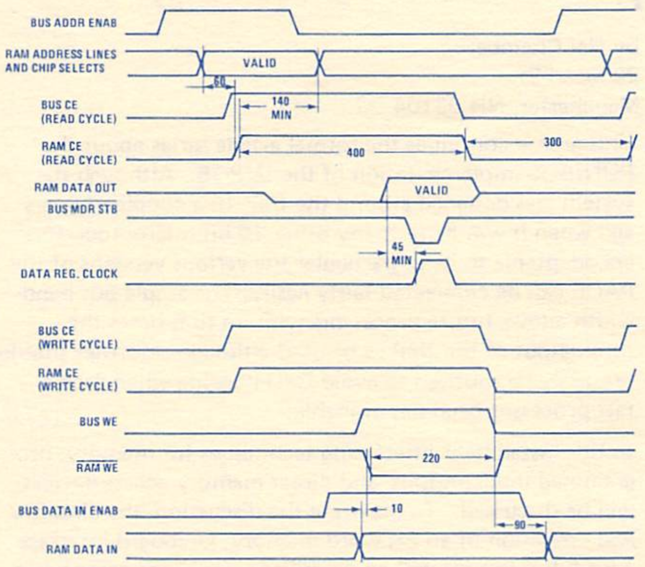


Figure 2. 8K Board Timing When Driven by Standard Bus Timing

Two on-board regulators are used to supply the +12 and -5 voltages needed by the RAM's. Negative 5 is simply produced with a zener diode powered from the BUS -15 line. This is allowable since the -5 drain is in the microamp range. Positive 12 is derived from BUS +15 with a descrete component regulator. This was done to improve the accuracy of the +12 supply to the RAM's. A trimming resistor in parallel with one of the regulation feedback resistors is chosen to set the output voltage within about 1%.

Data into the board from the lower 12 bits of the bus is buffered by 8098 (or 74368) tri-state drivers which are continuously enabled thus acting as high power inverters. The buffered bus data ties into the 12 address lines of all 32 RAM's as well as the data in pins of the least significant 24 RAM's. The high power inverters are needed to drive the heavy capacitance presented by massed address inputs. Pullup resistors may be added to these lines when driving Intel 2107A type RAM's but should be omitted otherwise since the larger voltage swing creates more noise which low threshold RAM's would be sensitive to.

Data from the upper 4 bus data bits is run through standard inverters into the remaining data-in pins and also the chip select and board address recognition circuitry. A four-input NAND gate for each group of 16 RAM's recognizes addresses within its 4K area as selected by address jumpers and directly drives the chip select input which is latched in the RAM's just as the address lines are. A portion of a 7437 functions as an OR of the two chip select drivers and feeds the "board addressed" latch. Note that "board addressed" is inhibited if an I/O address is on the bus. Note also that this latch is clocked by the trailing edge of BUS ADDRESS ENABLE to allow maximum time for the board addressed decision to set up at its D input.

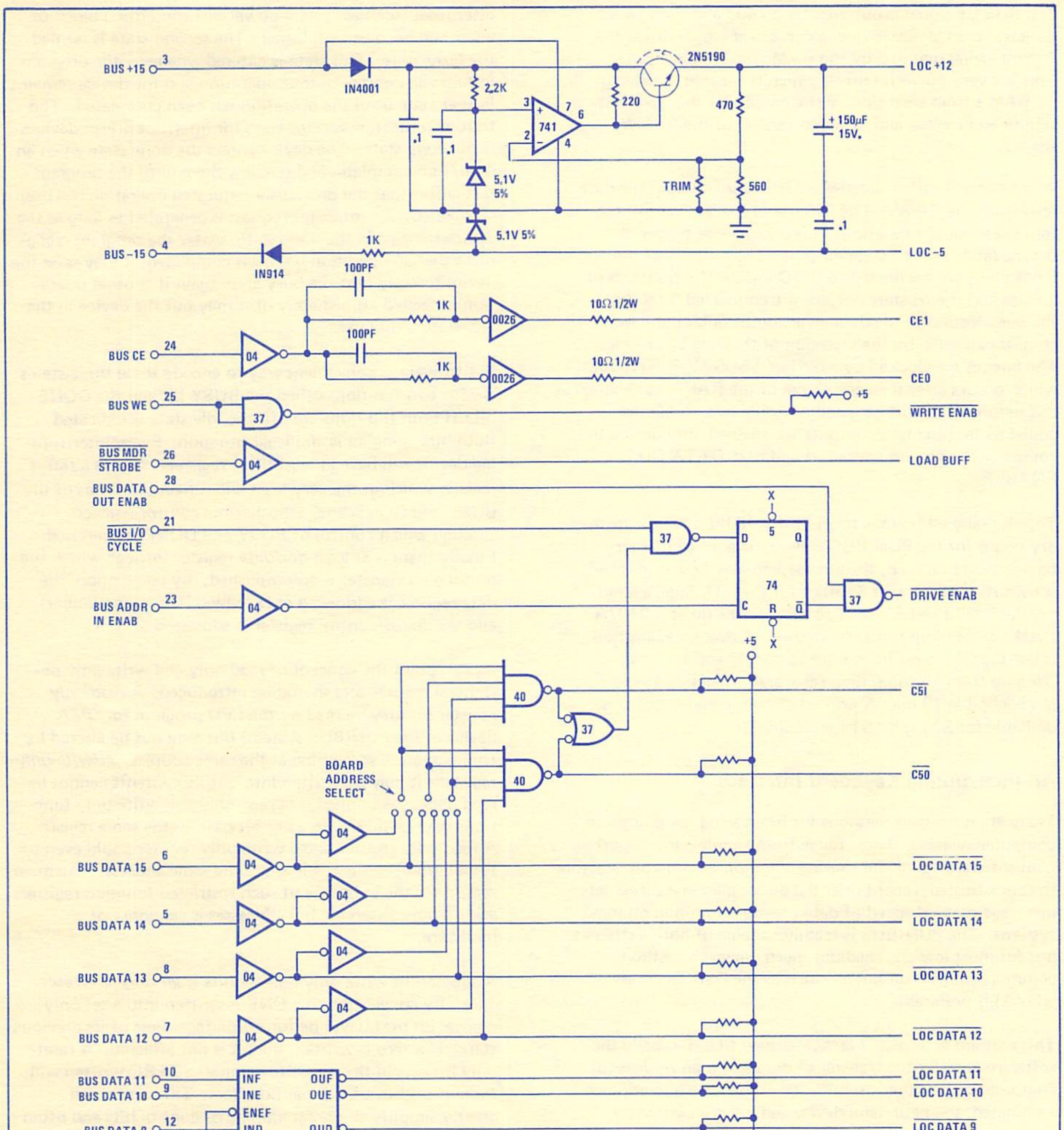


Figure 3. 8Kx16 Memory Control

The BUS CE signal is buffered by a 7404 and then goes directly to an MH0026 dual clock driver which drives the 12 volt swing required by the RAM chip enable inputs. This is a very powerful driver which is capable of driving 16 RAM's from each side. Write enable for the RAM's is simply an inverted and buffered version of the BUS WE signal.

As mentioned earlier, latches are necessary to hold the data read from the RAM's. The 8551 (74173) latch seems custom made for the function. It is a 4 bit edge-triggered D-type latch with tri-state outputs. The outputs of the RAM chips are connected to the D inputs of 4 packages of latches and the tri-state outputs are connected directly to the bus. Note that the data inversion through the RAM chips is cancelled by the inversion of the data bus receivers. The latches are clocked by a buffered BUS MDR STROBE which occurs after the access time of the RAM's has elapsed but before data must be gated onto the bus. A gate is added so that the latch outputs are enabled only during the coincidence of board addressed and BUS DATA OUT ENABLE.

That is really all there is to an 8K by 16 bit dynamic memory board for the PUNIBUS system. If a lot of memory boards are to be used, the bus loading may be reduced by substituting low-power Shottky for the TTL logic excluding the 74173 and 7440. This may not be done if 2107A RAM's and pullup resistors are used. Power consumption is less than 1.0 amp from the +15 supply and less than .25 amp from +5 (no pullup resistors). Negative 15 drain is a neglibible 10 ma. A printed circuit memory board is available for $30 (plus $1 for shipping).

An Interrupting Keyboard Interface

There are numerous methods for interfacing peripherals to computer systems. These range from simple parallel ports with software status monitoring common on microprocessors to sophisticated, record-oriented direct memory access setups (regardless of speed of data transfer) common on maxi systems. The PUNIBUS is readily capable of both extremes but for most low and medium speed devices a method common on minicomputers (such as the Data General NOVA) is preferable.

This method is termed *interrupt-driven* I/O. Basically the software *initiates* the transfer of data and then returns to computation or other useful work. When the operation is completed, the peripheral device generates an *interrupt* which temporarily diverts attention back to the peripheral device. The program's response to the interrupt may be either the initiation of another operation or simply an acknowledgment that the previously requested operation has been completed and that everything is OK. The main advantage of interrupt driven I/O is that simultaneous computation and operation of one or more I/O devices is possible with relatively simple programming procedures. The same effect can be achieved with a simple status checking loop in non-interrupt systems but the programming becomes specialized and cumbersome.

Basically every mechanical peripheral device (one that takes a relatively long time to process a character) has three "states" that it can be in. The first state is the *idle* state in which nothing is happening. This state would be entered after reset (such as system power-on) and after a block of work had been accomplished. The second state is termed the *busy* state. This state is entered whenever the program initiates an input or output operation and the device remains in that state until the operation has been completed. The third and most important state for interrupt driven devices is the *done* state. The device enters the done state when an operation completes and remains there until the program recognizes that the previously requested operation has been completed. An interrupt request is generated as long as the device remains in the done state. After the program recognizes that an operation has been completed, it may send the device directly into the busy state again if another operation is needed immediately or it may put the device in the idle state until later.

In hardware, a convenient way to encode these three states is with two flip-flops called the BUSY bit and the DONE bit. If both flip-flops are off, the idle state is indicated. Both bits being on is an illegal situation. Every interrrupting device will have at least three registers. One is a *status* register which at the very least will reflect the states of the BUSY and DONE bits. Another is a *control* register through which control of BUSY and DONE is exercised. Finally there is at least one *data* register through which the useful data transfer is accomplished. By convention, the data register is addressed at N (which is an even number) and the status/control register is addressed at N+1.

At this point the concept of read-only and write-only peripheral register bits should be introduced. A *read-only* register bit may be read by the CPU program (or DMA device in the PUNIBUS system) but may not be altered by trying to store something at the same address. A *write-only* register bit may be written into but its contents cannot be read back. As implied, the read-only and write-only functions may be mixed on a per bit basis in the same register. A read only register and a write-only register could even be totally unrelated and still share the same address. The main reason for the existence of such restricted function registers aside from conserving I/O addresses is economy of hardware.

A subclass of write-only register bits is *set-only* and *reset only*. By convention, if a ONE is written into a set-only register bit the bit will be forced on regardless of its previous state. If a zero is written, the bit is not affected. A reset-only bit acts in the opposite manner; a ZERO written will force it off but a ONE has no effect. This action can greatly simplify the programming of control bits and often eliminate the need for a full *read-write register.*

One final variation of a write-only register bit is a *pulse-generate* bit. When a ONE is written into a pulse-generate bit, a short pulse is generated which performs a specific function in the device. Nothing happens if a ZERO is written.

As a specific example, consider an alphanumeric keyboard and its interface. This is a mechanical device with great variance of data transfer rate and probably one that would benefit most from interrupt capability. Associated with the keyboard interface will be an 8 bit read-only data register and a four bit status and control register. The function of the data register is self explanatory so we will examine the status/control register in detail.

The status/control register consists of 4 significant bits. Bit 3 is a regular read/write bit and is called INTEN (INTerrupt ENable). If it is on, the keyboard operates in an interrupt mode. If it is off, it operates in a simple polled status mode. Two more bits are the busy and done bits. BUSY is bit 2 and is a set-only bit while DONE is bit 1 and is reset-only. These two bits may also be read to determine the keyboard status. Bit 0 actually performs two separate functions. When written into, it performs as a pulse-generate bit which issues a reset pulse whenever a ONE is written into it. The reset pulse resets the keyboard interface just like a power-up sequence would. When read, it indicates whether the last operation completed had some kind of error. Actually, these bit assignments of the status/control register are the same on all interrupting devices, not just the keyboard.

Now let's go through a typical keyboard operating sequence to get an idea of how these bits work together. Initially the keyboard is in the idle state after power-up or a reset signal. Both BUSY and DONE are off as is INTEN. Now let's assume that a program is entered which uses the keyboard on an interrupting basis. First the keyboard must be taken from the idle state to the busy state. This is accomplished by writing X'000C to the keyboard status and control register. Bit 2 being a ONE sets the BUSY bit and bit 3 being a ONE enables interrupts from the keyboard. Of course the master interrupt enable control flag on the IMP must also be set before interrupts from any device can be accepted. At this point the program may go off and do something else.

While the keyboard is in the busy state, an LED lamp is illuminated which tells the operator that a key may be struck. When a key is struck, the keyboard goes into the done state by resetting the BUSY bit and setting the DONE bit. Also the keyboard LED extinguishes. Now that DONE and keyboard INTEN are both on, an interrupt request is generated. The interrupt service routine looks at the DONE bit in the status word of every device that might have caused an interrupt to determine who the guilty party is. Assuming that the keyboard service routine gains control, the key code may be read from the keyboard data register. After the character is saved in memory or otherwise interpreted there are two possibilities. If no more characters are needed from the keyboard, the DONE bit may be turned off by sending a X'0008 to the keyboard status and control register which puts the keyboard back into the idle state. If more characters are needed, DONE may be turned off and BUSY simultaneously turned on by sending a X'000C to the keyboard. In either case clearing the DONE bit (by writing a ZERO to it) releases the interrupt request on the bus.

Non-interrupt operation may be accomplished by keeping the INTEN bit off and simply testing either the BUSY or the DONE bit in the status register repeatedly. If an input operation is started and for some reason it needs to be aborted, the reset pulse may be generated by sending a X'0001 to the keyboard control register.

Now let's take a look at an actual keyboard interface circuit. In figure 4 is shown an interface for a Clare-Pendar keyboard which is a popular yet reasonable priced keyboard. It features full ASCII coding with upper and lower case. Besides a standard shift lock, it also has an "upper case lock" key which affects only the alphabetics when locked so that program source text may be easily typed using a dual case keyboard. Several extra keys are present which are simply unencoded normally open contacts.

Operation of the keyboard is very simple. Whenever a key is pressed, the key code is updated on 7 parallel output lines and simultaneously a 20 millisecond positive going strobe pulse is generated. The new code is valid on the falling edge of the strobe.

Keep in mind when going through the interface logic operation that the general principles can be applied to interfacing any programmed I/O device to the PUNIBUS system. Starting at the bottom of the page, a 7430 and associated inverters detect the presence of a keyboard address, either X'FF00 or X'FF01. Remember in the bus description that the $\overline{\text{BUS I/O ADDR}}$ line goes active low whenever the bus data lines contain a binary number between X'FF00 and X'FF7F. The least significant bit determines which of the two 7474 flip-flops will be set if the 7430 is satisfied. The flip-flops are needed to hold the "keyboard addressed" information after the address disappears from the bus. The flip-flops are clocked by the trailing edge of BUS ADDR ENAB in order to allow maximum time for the address decoding. This is permissible as long as the logic path between BAE on the backplane and the flops has only one or two gate delays.

Taking the simplest case first, the bottom flip-flop is turned on when address X'FF01 is detected which is the keyboard data register address. A set of tri-state buffers at the top of the drawing are turned on during the coincidence of this flop being set and BUS DATA OUT ENAB. This is really all that is required to interface with any kind of simple input data source. A similar connection to the top flip-flop allows reading of the keyboard status but only 4 bits are significant. Any bus lines not driven during a read cycle will be read as ones.

Note that a single-shot for driving a small speaker on the keyboard is triggered whenever the keyboard data register is read. This gives valuable operator feedback and prevents undetected double entries. Also, various beep tones may be programmed with timed loops.

Now for the fun part, the BUSY and DONE flip-flops and the interrupt circuitry. Really it is all very straightforward with one flip-flop for each of the functions of BUSY, DONE, and INTEN. Due to the inverting bus data receivers these flip-flops are all upside-down, that is, they are on when reset and off when set. The following explanation will be given as if they were rightside-up which would have required 3 more inverters to actually implement.

First note that all of the flip-flops may be reset by the $\overline{\text{BUS RESET}}$ signal. They may also be reset by the coincidence of keyboard status and control address (upper address flop on,) BUS WRITE ENAB being true and BUS DATA 0 being a ONE as detected by the 74LS10 gate. Actually this general structure will be found on any write-only pulse generate type of output.

Next note that all 3 flip-flops are clocked whenever the status/control register is written into as determined by the coincidence of status/control addressed and BUS WRITE

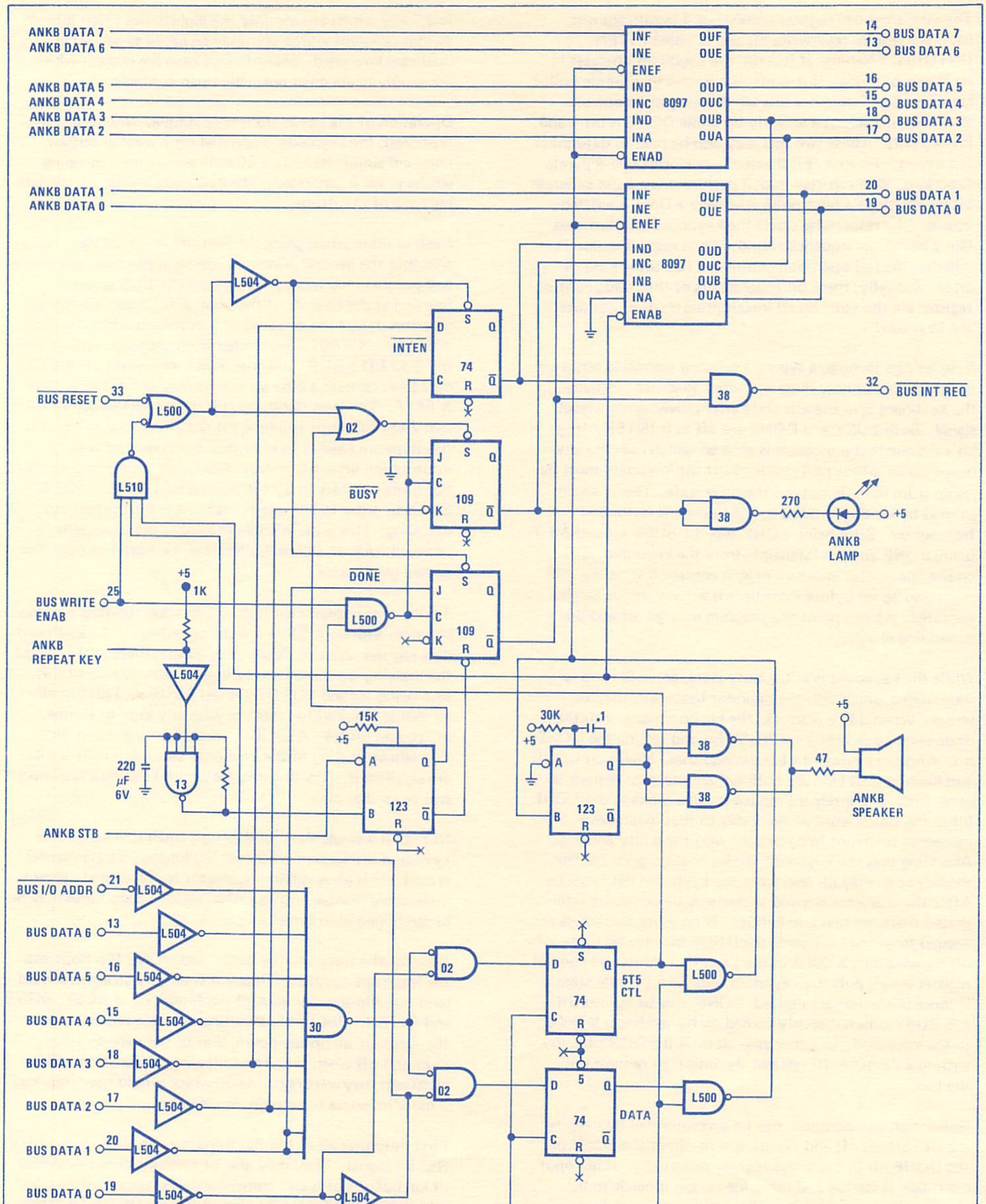


Figure 4. Interrupt Keyboard Interface

ENAB. Since INTEN is a D-type flip-flop, it is updated according to BUS DATA 3 whenever the control register is written into. BUSY, being a J$\overline{K}$ type with the $\overline{K}$ input deactivated, can only be set by a ONE on BUS DATA 2 when it is clocked. A ZERO under these conditions would do nothing since both J and $\overline{K}$ would then be inactive. This is a general example of a set-only peripheral bit. DONE is wired in the opposite manner with its J input tied off and its $\overline{K}$ input connected to BUS DATA 1. ONES written to it do nothing while ZEROES will reset it and so it is a reset-only peripheral bit. Note that the 74109 JK flip-flop seems custom made for this use.

When a key is struck a single-shot detects the trailing edge of the keypressed strobe and generates a short pulse (about 100NS) in response. This pulse directly resets BUSY through the 7402 and directly sets DONE through its preset input thus implementing the description given earlier. The same single-shot may also be triggered by the repeat oscillator which is activated when the unencoded "RPT" key is pressed. Since a register on the keyboard holds the key-code of the last key pressed, that code will be repeatedly sent. The repeat rate is about 10 per second but it may be changed by altering the 220 μF capacitor in the oscillator.

The interrupt circuit is simplicity itself. A single 2-input open-collector gate pulls down $\overline{\text{BUS INTERRUPT REQUEST}}$ whenever INTEN and DONE are simultaneously on. The interrupt remains active until the interrupt service routine resets DONE by either sending the keyboard to the busy state for another character or sending it to the idle state. TO BE CONTINUED.

SOFTWARE updates

PCASM

For the PACE Conversational Assembler (PCASM) part number 4400035, revision B, the following change should be made:

Address	Data
0A43	01900

UDS

Programs linked with revision B of LINKEDIT can only be loaded with revision C of the UDS (PACEDOS) firmware. Revision C of the UDS diskette has revision B of LINKEDIT on it; so you must have revision C firmware to use the revision C UDSdiskette. If you got your DOS prior to 10/12/77 and you wnat to use revision B LINKEDIT (which allows loading into addresses > 32K, e.g., split base page), you must update your firmware to revision C. Order no. IPC-16S/903M $395 includes 8 ROMS, 2 diskettes, BASIC and a set of PACE manuals. This can be ordered from your nearest distributor. A Pace Firmware update Kit is available from the Microprocess or Service Center (order no. 935305556-001). It includes 8 ROMS and costs $160.00.

UDS and PACE UPDATE

For users with a PACE Development System and wish to run at higher baud rates with the current loop interface without the Disk/CRT interface card, the following are the recommended modifications to be made to the PACTTY (PACE Teletype I/O Driver Routines – Rev. B) firmware.

For higher baud rates with even parity correction, the modifications necessary are,

300 Baud	(7E9C) from 4603 to 4600
1200 Baud	(7E9C) from 4603 to 4600, and (7E9E) from 4401 to 4400

For higher baud rates with no parity corrections, the modifications necessary are,

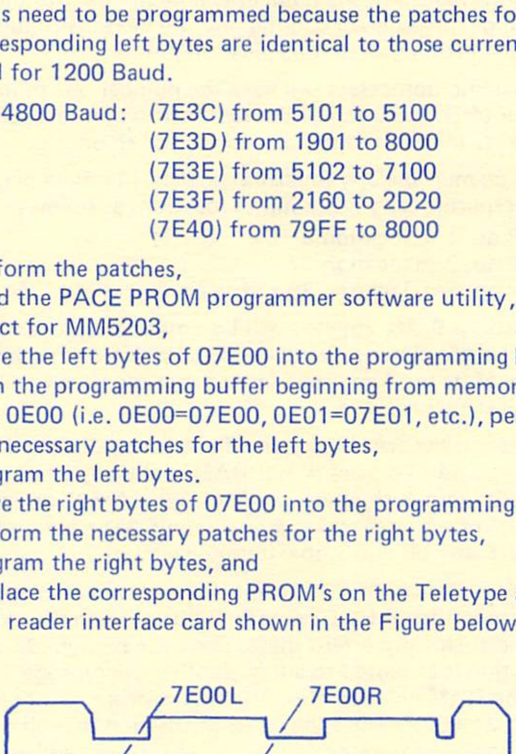
300 Baud	(7E4E) from 5106 to 5102, and (7E50) from 510C to 5104, and (7E54) from 2DC0 to 2D8E
1200 Baud	(7E4E) from 5106 to 5101, and (7E50) from 510C to 5102, and (7E51) from 21F0 to 2160, and (7E54) from 2DC0 to 2D20

Universal Development System users may replace their current 1200 Baud maximum console data transfer rate with either 2400 or 4800 Baud by making the following firmware patches to revision B of PACTTY, the PACE Teletype I/O Firmware Routines,

1. For 2400 Baud: (7E3F) from 2160 to 2120
   (7E42) from 2D20 to 2D16

   Note that only a PROM containing the data for the right bytes need to be programmed because the patches for the corresponding left bytes are identical to those currently used for 1200 Baud.
2. For 4800 Baud: (7E3C) from 5101 to 5100
   (7E3D) from 1901 to 8000
   (7E3E) from 5102 to 7100
   (7E3F) from 2160 to 2D20
   (7E40) from 79FF to 8000

To perform the patches,
1. Load the PACE PROM programmer software utility,
2. Select for MM5203,
3. Move the left bytes of 07E00 into the programming buffer,
4. With the programming buffer beginning from memory location 0E00 (i.e. 0E00=07E00, 0E01=07E01, etc.), perform the necessary patches for the left bytes,
5. Program the left bytes.
6. Move the right bytes of 07E00 into the programming buffer,
7. Perform the necessary patches for the right bytes,
8. Program the right bytes, and
9. Replace the corresponding PROM's on the Teletype and card reader interface card shown in the Figure below.



multi-microprocessor software

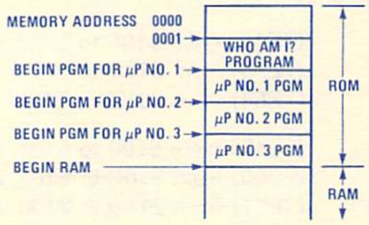
By Dan Grove, FAE

Introduction

This application note describes a software approach to allow individual microprocessors to access the proper program in ROM after power or reset.

The problem in multi-microprocessor systems is to get each microprocessor's program counter to access the program assigned to that microprocessor with power on or reset. Each SC/MP microprocessor will, by design, go to location X'0001 to access its first instruction. Therefore, location X'0001 in ROM must contain a small program which will allow each microprocessor to determine, in sequence, who it is and direct its program counter to the appropriate location in ROM.

The ROM may look like this:



The microprocessors will use the ILD (Increment and Load) instruction to determine who they are. This instruction requires a RAM location and the fact that the RAM location *MUST* be zero after power on.

Assuming that the RAM location is guaranteed to be zero after power on, then the "Who Am I" program will operate like this.

Each microprocessor will increment the contents of the RAM location and load that value into its accumulator. The microprocessor will then multiply this unique number by a constant. After multiplication the number will be placed into the lower portion of the program counter and the next instruction will be fetched from that unique location plus one.

The first microprocessor will have the number one in its accumulator after execution of the ILD instruction, the second number two, the third number three, and so on.

If each μp multiplies by the same constant (for example, eight) the microprocessors will jump to locations as follows:

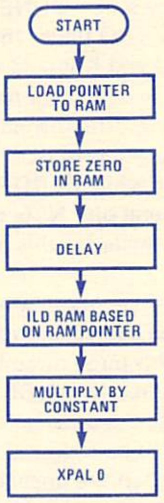
μP no. 1 to location 9 (PC = 8 + 1)
μP no. 2 to location 17
μP no. 3 to location 25 and so on

At locations 9, 16, 25, etc., will be similar programs which instruct each microprocessor to load a pointer with the address of its PGM and the jump to that program via an XPPC instruction.

We assumed the RAM location will be zero when the first microprocessor accesses it, but RAM is obviously not guaranteed to be zero with power on. The "Who Am I" program must then first somehow guarantee the RAM location is zero *BEFORE* any μP uses it to determine who he is.

The first section of the "Who Am I" program will instruct the microprocessors to load a pointer pointing to the RAM location and then store a zero there. Then a maximum delay instruction is executed to allow all other microprocessors to clear the RAM location also. The first microprocessor will be assured when it finishes the delay instruction that all other microprocessors have zeroed the RAM location and are somewhere behind it in the delay instruction. When the first microprocessor exits the delay instruction it will then perform the sequence beginning with ILD as described above.

The flow chart for the complete "Who Am I" program will look as follows:



A typical program may look like this:

```
0001  LDI X'10    ; Set up
      XPAH 2      ; RAM pointer
      LDI 0       ; Store zero
      ST (2)      ; in RAM
      DLY 255     ; Wait until all μP have been stored
                  ; zero
      ILD (2)     ; Increment RAM and put copy
                  ; into accumulator
      RR          ; Multiply by
      RR          ; a constant
      RR          ; eight
      RR
      RR
      XPAL 0      ; Put number into lower
                  ; program counter so the
                  ; next instruction is
                  ; fetched from location
                  ; N + 1
```

At locations 9, 17, 25, etc., the programs may look similar to this:

```
0009  LDI XNN
      XPAH 2
      LDI X'MM
      XPAL 2
      XPPC 2
0011  LDI X'UU
      XPAH 2
      LDI X'VV
      XPAL 2
      XPPC 2
```

where X'NNMM and X'UUVV are the beginning addresses of programs minus one for microprocessors one and two.

This "Who Am I" program is usable for multi-microprocessor systems with up to sixteen microprocessors if the constant eight is used.

To guarantee that microprocessor no. 1 does in fact get program no. 1 and so on, a simple delay circuit is connected to each reset pin as shown. The RC time constant is chosen so that each microprocessor is held reset 1 ms longer than the previous one.

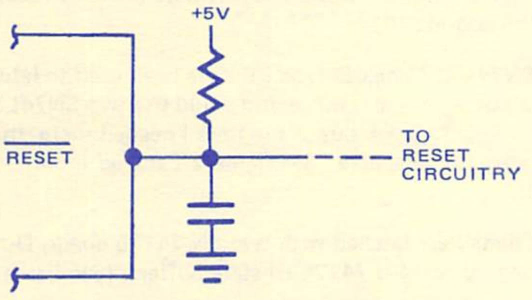
Now each microprocessor will enter location X'0001 in ROM 1 ms apart, clear RAM, delay ¼ sec, ILD RAM, multiply RAM value by a constant, jump to a unique program that loads a pointer with a unique address, and then jump to its particular program based on that address. Eight is an ideal number to multiply by since it optimizes the use of ROM (at locations 9, 17, 25, etc.) by having programs that load each μP's program counter packed contiguously at the end of the "Who Am I" program.

This is good, though only if the "Who Am I" program is less than eight bytes long; unfortunately it is greater than eight bytes long and overlaps from location 9 on up.

This can easily be remedied by offsetting locations 9, 17, 25, etc., by a constant with the add immediate instruction.

If we wan. locations 9, 17, 25, etc., to be 20, 28, 36, etc., we simply add immediately decimal eleven just before the XPAL 0 instruction.

Now the program counter loading programs are ideally spaced with relation to each other and the "Who Am I" program in ROM.



The final program may look as follows:

```
Memory
Address
 0001   LDI X'10     ; "Who Am I"
        XPAH 2
        LDI 0
        ST (2)
        DLY 255
        ILD (2)
        RR
        RR
        RR
        RR
        RR
        ADI 11       ; Add offset of
        XPAL 0       ; eleven
 0014   LDI X'NN     ; Load up no. 1 PC
        XPAH 2       ; with pointer to
        LDI X'MM     ; its program
        XPAL 2
        XPPC 2
 001C   LDI X'UU     ; Load up no. 2 PC
        XPAH 2       ; with pointer to
        LDI X'W      ; its program
        XPAL 2
        XPPC
 0024   LDI . . .
        . . .
        . . .
        . . .
```

downloader

By Keith Winter and Milt Schwartz

Introduction

This application note describes a simple interface between National Semiconductor's Universal Development System and the SC/MP LCDS for the purpose of down-loading object modules. The output of the PACE-SC/MP Cross Assembler (PSCA12) from the UDS results in a transfer rate of 110 baud to a Teletype interface.

Operation

The Teletype initially is connected to the SC/MP LCDS to allow the user to put the LCDS into the paper tape load mode. The Teletype is then disconnected from the LCDS and the UDS is connected in place of the Teletype. Using the UDS CRT, source code is assembled from disc specifying the Teletype as the device to receive the object module. When the assembler halts, select 110 baud, push RUN on the UDS Control Panel and the object code will be transferred into LCDS RAM.

Detailed Description

Ensure that LCDS RAM occupies the same address location as specified by the user program.

Connect the Teletype to the LCDS, enter the Load Command (L (CR)) and then disconnect the Teletype from the LCDS.

Referring to figure 1, connect pin 1 and pin 13 on the UDS Teletype connector to the LCDS rear terminal strip labeled GND. Connect pin 4 on the UDS Teletype connector to the LCDS rear terminal strip position labeled + TTY XMIT.

Using the CRT as the input terminal to the UDS, assemble the source code and specify the object module to be output to the Teletype.

Example: ASM – <u>I=FILENAME,NL,0=*PT (CR)</u>

Note: Underlined characters are user input.

The PACE-SC/MP Cross Assembler completes pass 1, prints the message "TURN ON PT PUNCH AND PRESS RUN," and then halts.

Select 110 baud by simultaneously depressing the AUX 1 and AUX 2 switches on the UDS Front Panel.

Depress the RUN switch. The assembler will execute pass 3 and then transmit the object code to the LCDS via the Teletype output port. After pass 3 is complete, the UDS will halt. Disconnect the wires between the LCDS and the UDS and reconnect the Teletype to the LCDS. The program in the LCDS RAM can now be executed using the GO command (G (CR)).

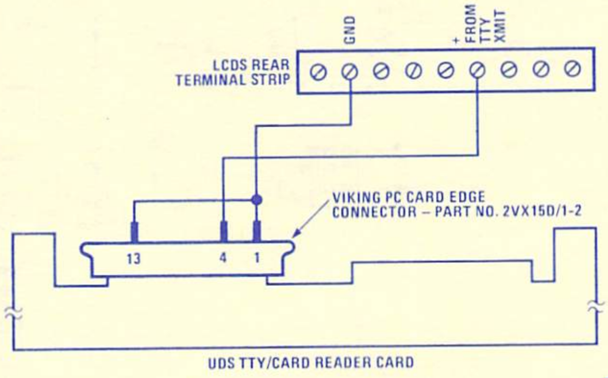


Figure 1. Downloader

SC/MP Introkit and Keyboard Kit Expansion

by Albert Schweitzerlaan 41
Vleuten
Holland

As I developed some SC/MP controlled machinery I found that a major disadvantage in using the Introkit and Keyboard kit is that, except for the hexadecimal display on the keyboard, no outputs are available.

The modifications I made to the Introkit and Keyboard kits provide the user with the following:

- An 8-bit parallel data I/O at TTL level; (output LS, input normal TTL)
- Output may be cleared independent of MPU.
- Output signals on the flag lines F0, F1 and F2 as well as signals on the output have been made visible with 11 LEDs.
- Several SR flipflop configurations have been used to provide bounce-free SA, SB, CONT, NRST and Clear to output signals.

The description can be rather short because of the simplicity of the circuit diagrams.

Address decoding for the ROM, RAM, display and I/O is done by the circuit on page 1; addresses are respectively:

```
ROM  *00* **** ****     display  *10* **** ****
RAM  *11* **** ****     I/O      *01* **** ****
```

In the addresses mentioned above, address bits AD12 to AD15 have *not* been used; * mean don't cares.

Address decoding is done on lines AD09 and AD10 only, so only the significant values for these lines have been shown. (See figure 1.)

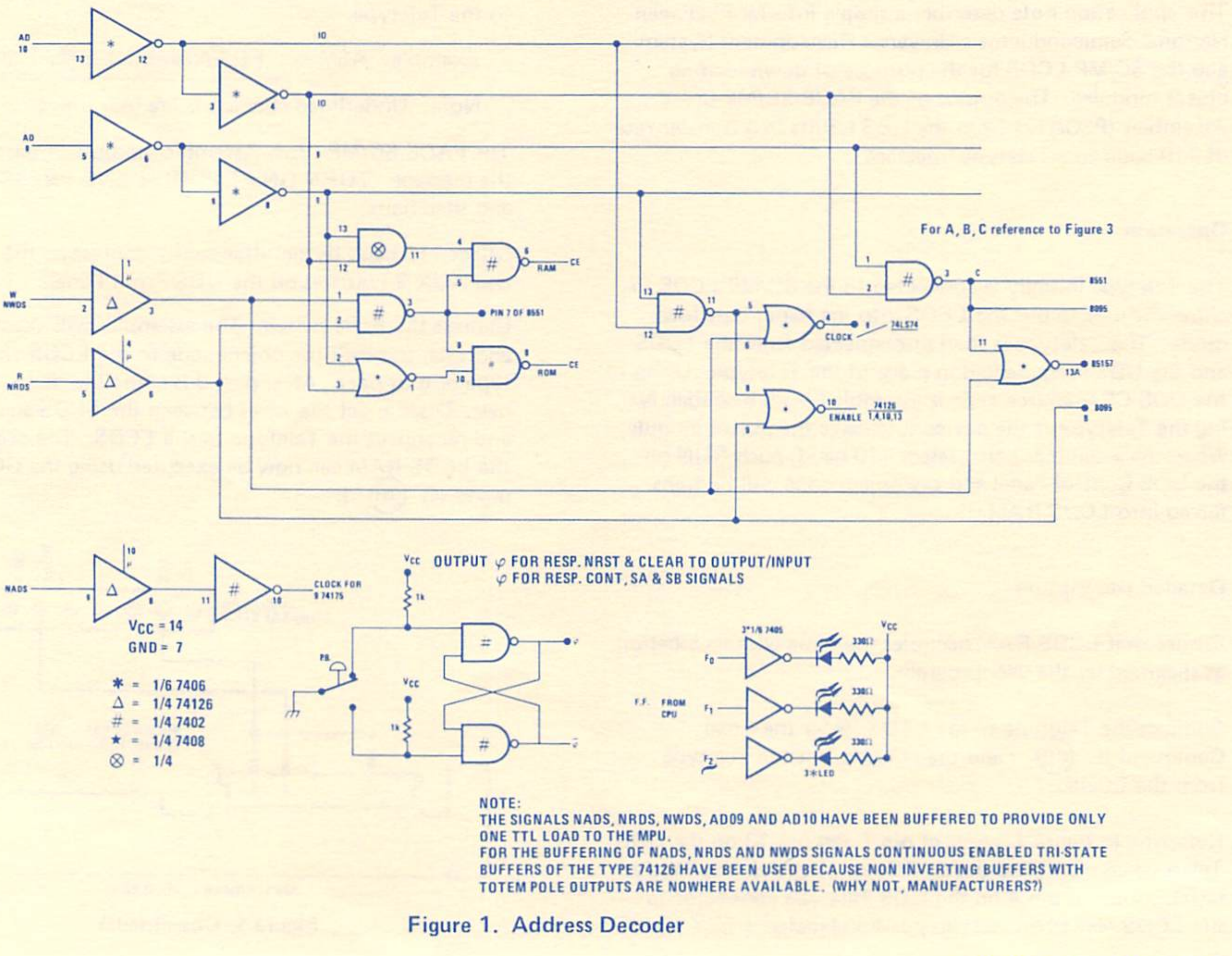
Output and input are treated as if it were memory locations with the address *01* **** ****

Two SN74LS174 hex D-type FF have been used to latch the output signals; of course you could use two SN74LS175 quadruple D-type FF but at the time I needed them, those parts were not available. See figure 2 Latched Input/Output Signals.

Input signals are latched with two SN 74175 quadr. D-type FF followed by two SN 74126 tri-state buffers. (See figure 2.)

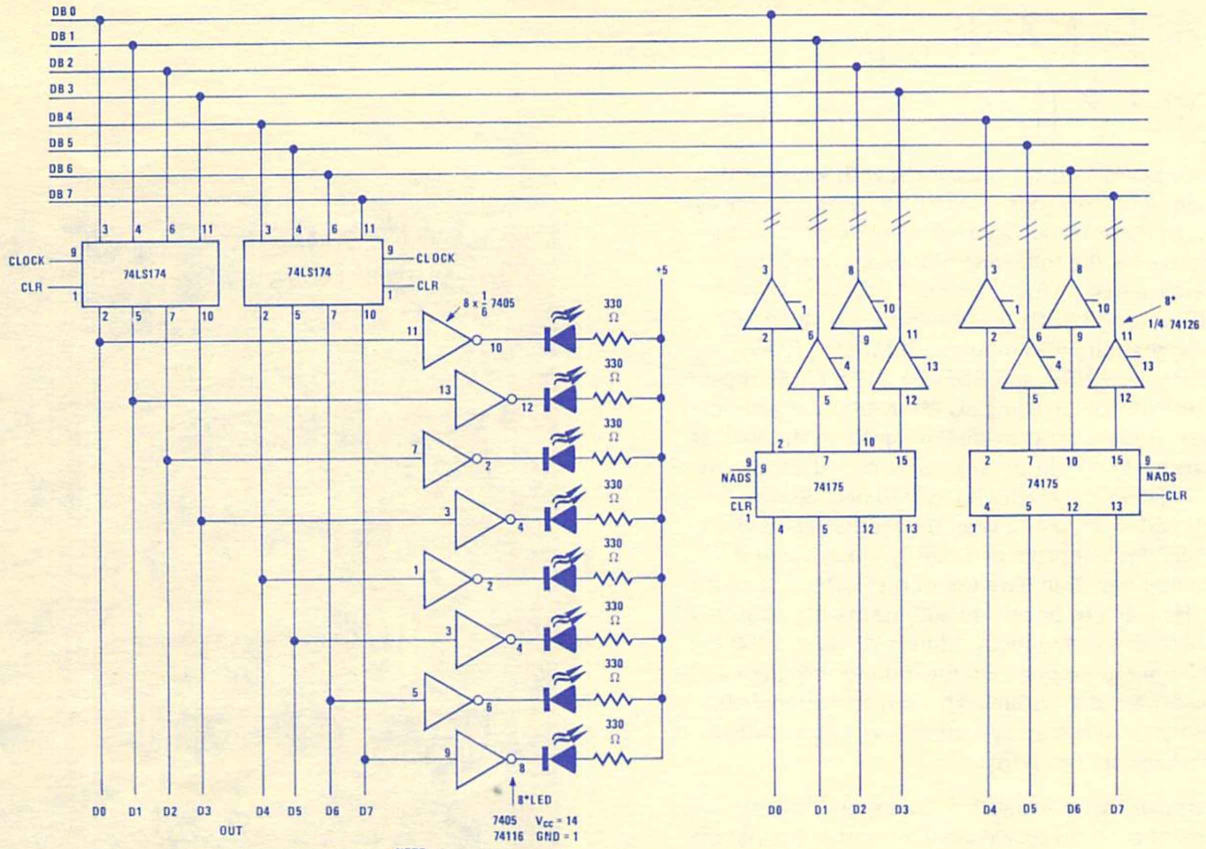
Clock signal for the input is the NADS signal followed by the NRDS signal so input signals to the MPU are always stable.

Output clocking is directly derived from NWDS signals.



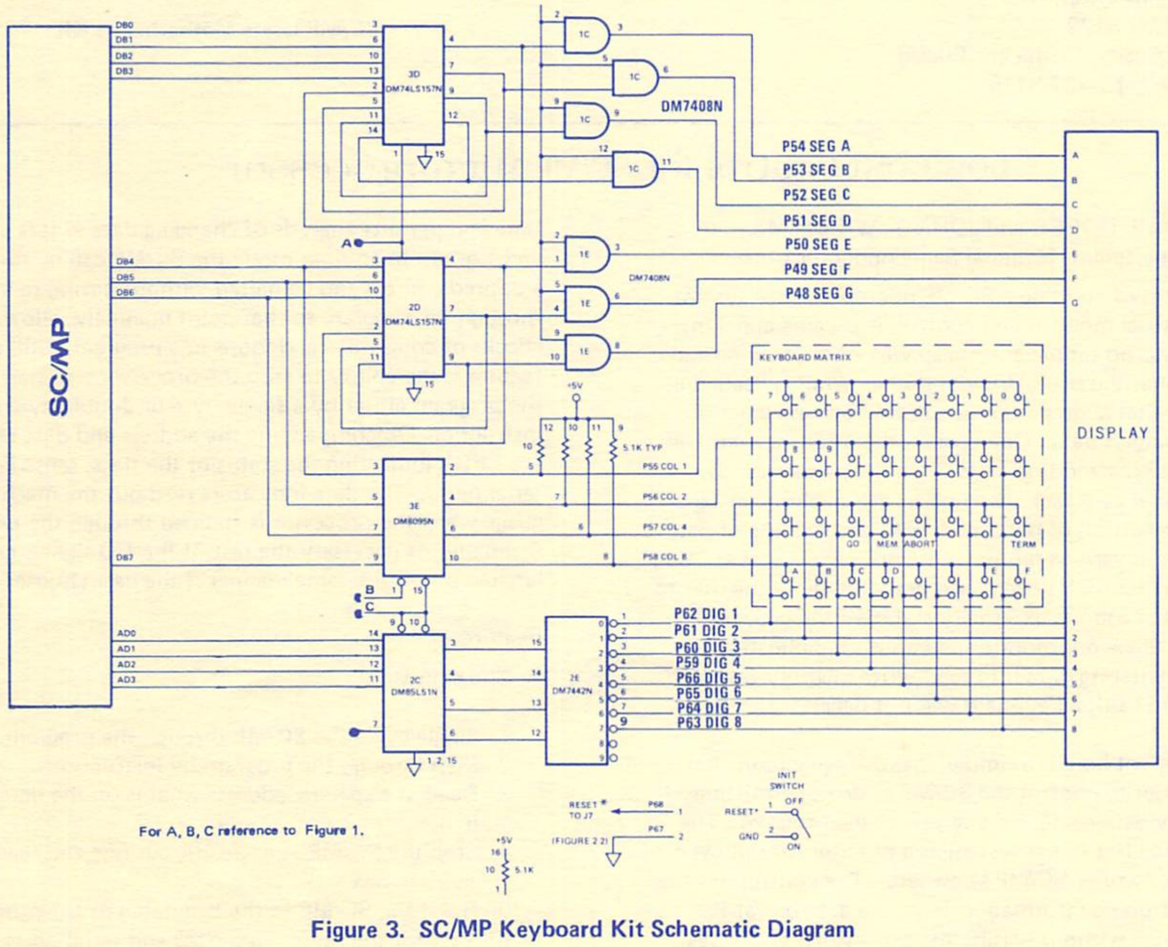
NOTE:
THE SIGNALS NADS, NRDS, NWDS, AD09 AND AD10 HAVE BEEN BUFFERED TO PROVIDE ONLY ONE TTL LOAD TO THE MPU.
FOR THE BUFFERING OF NADS, NRDS AND NWDS SIGNALS CONTINUOUS ENABLED TRI-STATE BUFFERS OF THE TYPE 74126 HAVE BEEN USED BECAUSE NON INVERTING BUFFERS WITH TOTEM POLE OUTPUTS ARE NOWHERE AVAILABLE. (WHY NOT, MANUFACTURERS?)

Figure 1. Address Decoder



NOTE:
AS 74LS174 IC'S HAVE BEEN USED TO LATCH OUTPUT SIGNALS CARE MUST BE TAKEN NOT TO OVERLOAD THE OUTPUTS. REMEMBER: ONE TTL LOAD TO OUTPUTS HAS ALREADY BEEN USED TO MAKE OUTPUT SIGNALS VISIBLE.

Figure 2. Latched Input/Output Signals



For A, B, C reference to Figure 1.

Figure 3. SC/MP Keyboard Kit Schematic Diagram

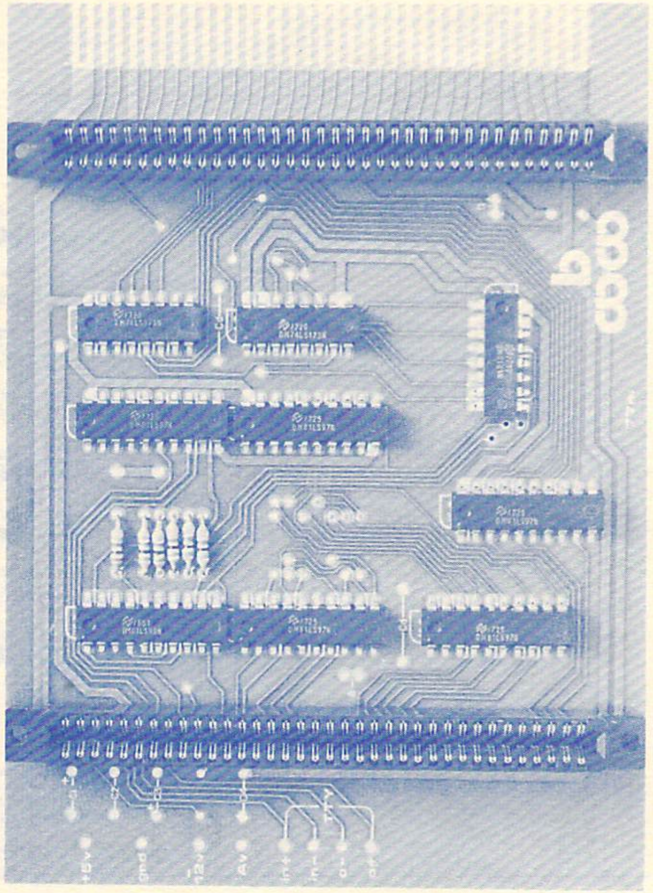
SC/MP Mother-Board Kit

Expand your SC/MP Kit (or SC/MP Kit with keyboard) into a system capable of addressing FFFF bytes of memory. In addition to these 16 buffered address lines, our Motherboard Kit provides the following buffered output lines: BWDS, BRDS, BADS, READ, FETCH, DELAY, HALT, BFLG1, BFLG2, and BSOUT. Provisions are also made for buffering the following inputs: MEMRDY, RUN, ENCPU, BSIN, ASENSE, and BSENSE. Our Motherboard Kit makes use of your existing SC/MP Kit by the addition of 32 wires. These wires provide the signals to the Motherboard necessary to obtain the signals described above. In operation, logic on the Motherboard will decode the address bits and when bit 12 is on the Motherboard will disable the SC/MP Kit prom and ram (and keyboard if used). The user will then have use of his SC/MP Kit board as before. He will also be able to address memory and I/O boards answering to any unique address between 1000 to FFFF. The only limitation is in the monitor program available on the SC/MP Kit system. This can be overcome by loading into your extended system ram a more complete monitor package or even NIBL.

The Motherboard Kit provides for one expanded output that consists of a 72-pin edge-connector that is compatible with many of the SC/MP related boards currently available. The SC/MP Motherboard Kit sells for $99.00 plus $2.00 shipping and handling.

For more information contact:

BW Ward Enterprises
P.O. Box 8122
Long Beach, California 90808
Phone (213)427-3418


SC/MP Users Motherboard Kit

Low Co$t Control for SC/MP Microprocessor

by GEORGE J. HOFER and JOHN T. WILLIAMS, Ion Implant Department, National Semiconductor Corp.

In the process of building a SC/MP microprocessor system for an industrial monitor and control, it became apparent that there was no simple and cheap way of controlling and analyzing what the processor was doing. What is needed is a controller that allows the user to find how and when the processor handles data. One option, the LCDS system, did not satisfy all our requirements. Requirements such as allowing actual card rack insertion in the machine, instantaneous observation of port conditions and reading of data on the buses at various points in the program as it was being executed. The LCDS system also does not allow the user to enter a certain address externally and have the processor stop there. Since our monitor and control application did not call for entering data into read/write memory (as the LCDS system can), a keyboard was not necessary.

What we present here is a simple, inexpensive circuit that allows the user to control the SC/MP system, permitting fast and easy debugging and analysis of the program. The control can be left in the system as a monitor, or removed and used on another SC/MP elsewhere. The control has the ability of stripping information from the data bus at the desired address without halting the processor. This saves time and permits analysis of changing data as it is gathered and stored. In another mode the SC/MP can be run up to a desired address and be halted without having to step through the program to that point manually, allowing fast checks of conditions and loops in a program. Still another feature is the ability to stop the processor and step through the program either by a single byte or double byte of instruction. Accompanying the address and data read outs are LED's indicating the status of the flags, sense lines and serial ports. The data indicators read out the machine language when the processor is stepped through the program manually. If necessary the rest of the I/O signals can be latched out in the same manner if the user chooses to do so.

Features:

A. The ability to:

1. Single cycle the SC/MP through the program.
2. Step through the program by instructions.
3. Read at a specific address what is on the data bus in hex.
4. Stop the SC/MP at a specific address and read what is there in hex.
5. Reset the SC/MP to the beginning of the program.
6. Monitor the sense lines, flags and serial ports.

B. The control:

1. Can be made at low expense on a single processor board and plugged into a system.

Modes of operation and switch positions necessary to execute them:

A. *Run and monitor mode*: Switches = run, read-at.

This permits the SC/MP to run through the program without interruption and still monitor the data, flag sense and serial port lines. This is the normal mode of operation for the control.

B. *Read-at mode*: Switches = run, read-at, address switches.

With the processor running, punch up the address of the desired location on the address switches and monitor the data at that location while the processor executes the program. This mode can also be used for the single-cycle and single-instruction operations.

C. *Stop-at mode*: Switches = run, stop-at, address switches, start/step.

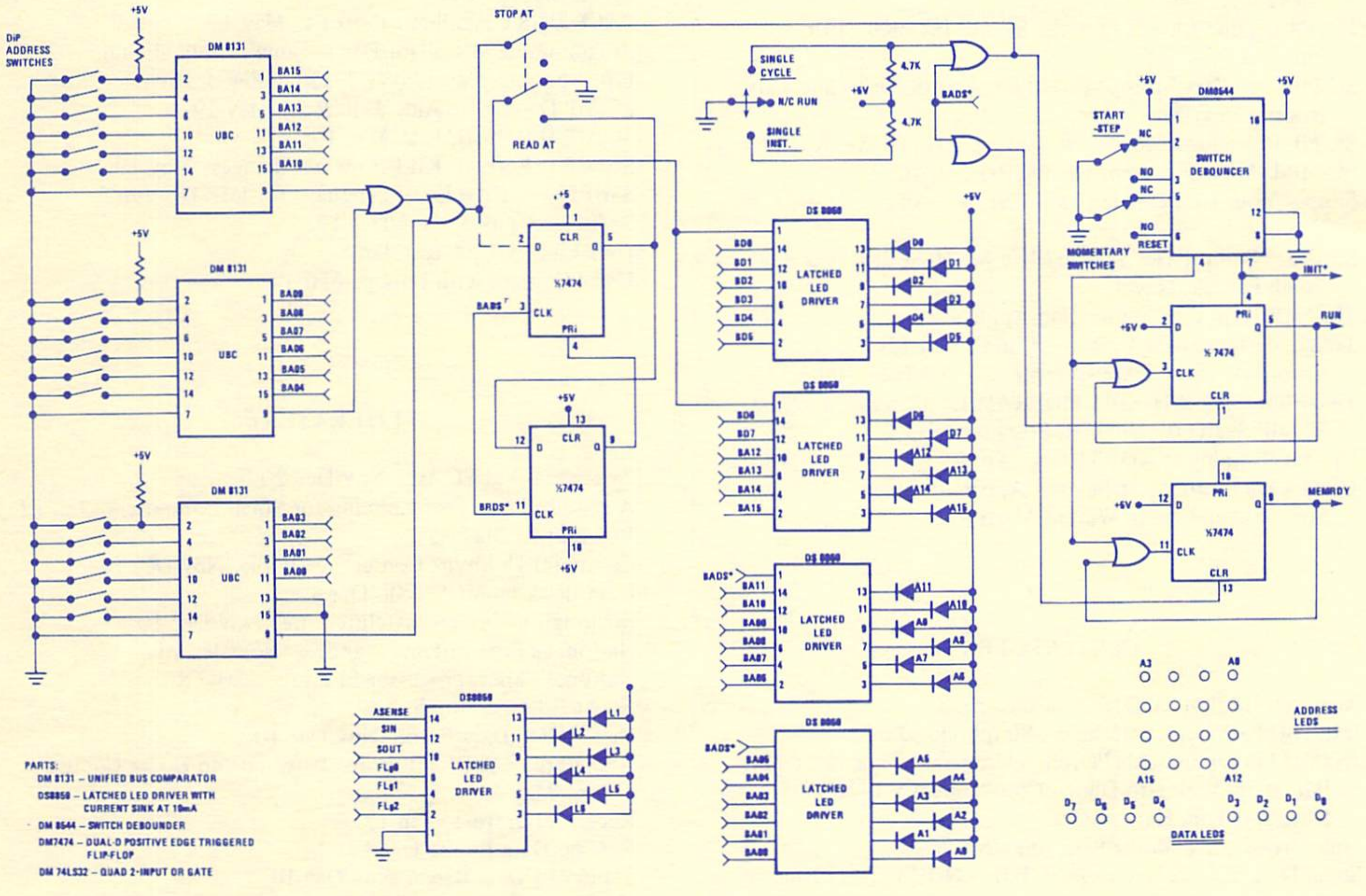
The desired address of the stop is entered on the address switches and the Stop-at/Read-at switch put to the Stop-at position causes the processor to execute the program up to the desired address, where the processor halts. To continue on, put the switch to Read-at and depress the Start/Step switch. This mode can be used in conjunction with the single-cycle and single-instruction operation modes.

D. *Single-cycle mode*: Switches = single cycle, read-at/stop-at, start/step, reset.

This mode allows the user to step through the program by a single byte of the instruction. To start at the beginning of the program have the switch in single cycle, and punch the restart switch. Then to-step through the program depress the start/step switch for each byte of instruction. The data indicators will now read out, in hex, the machine instruction for that address. To step up to a desired address select the particular address on the address switches and select stop-at. The processor will step up to that address and halt. To continue switch to read-at.

E. *Single-instruction mode*: Switches = single-instruction, read-at/stop-at, start/step, reset.

This mode has the same features as the single-cycle mode except that the steps are by instructions whether they are single or double byte instructions.



PARTS:
DM 8131 – UNIFIED BUS COMPARATOR
DS8858 – LATCHED LED DRIVER WITH CURRENT SINK AT 10mA
DM 8544 – SWITCH DEBOUNDER
DM7474 – DUAL-D POSITIVE EDGE TRIGGERED FLIP-FLOP
DM 74LS32 – QUAD 2-INPUT OR GATE

SC/MP Control

1977 compute index

ARTICLES

Add a Programmable I/O Interface to your SC/MP Kit. Thomas M. Farr Jr., May 1
ASCII-to-Baudot Code Converter, Leon Minton, Nov/Dec 20
A PROM Programmer for the SC/MP LCDS. Fran Terry and Keith Winter, Jul/Aug 1
A Real-Time Frequency Clock for PACER. J. R. Millenson, Jan 1
Bringing Up a SCAMP. C. R. Britten, Oct 1
Build the Coffee Can Special Eraser. Apr 2
CRIMP. Tom Harper, May 1
Homebrew SC/MP System Doubles as a Ham Repeater Scanner. Thomas F. Lewis, Sep 10
How to Expand a CMOS Key Encoder. Mar 5
INS8060 Meets TVT II. Soo Nam Kim, Nov/Dec 14
IMP-16P Programs MM1702's. Barney Hordos, Nov/Dec 11
Is Your Assembly Language Program Well Structured? Webb Simmons, Nov/Dec 18
Logical Methods to Get a PACE Extended Instruction Set Cheap. James F. Derry, Oct 6
Microprocessor Security Systems Made Easy. Phil Roybal, Feb 5
MM52Ø4Q EPROM Programmer. Leon Minton, Sep 6
NIBL – Tiny BASIC for National's SC/MP Kit. Mark Alexander, Feb 1
PROM Erasure Checker for the SC/MP IntroKit. Eric Skovgaard, Sep 9
SC/MP – A Very Musical Processor. Barny Hiney and Lars Rogers, Nov/Dec 2
SC/MP Homebrew Computer System. Dan Grove, Apr 11
SC/MP-II Microprocessor Highlights. Jul/Aug 13
Simply Your Homemade Assembler. Gregory C. Jewell, Feb 8
Software Setup Eases Traffic Flow for Microprocessors. Janak Pathak, May 4
The IMP-16 in Communication Applications. Apr 3
Tough Mathematical Tasks are Child's Play for Number Cruncher. Alan J. Weissberger and Ted Toal, Jun 6
Two Simple, Fast Hexadecimal RAM Loaders for Small SC/MP Systems. Dr. J. R. Stockton, Sep 1
Up and Running at NORTRON. Apr 9
Using a Microprocessor Beyond Apparent Speed. Milt Schwartz and Keith Winter, Mar 1

CENTERFOLDS

Analog to Digital (A/D) Converters. Feb
IMP-16C Peripheral Interfacing Simplified. Jan
INS8255 Programmable Peripheral Interface Data Sheet. May
SC/MP as an Analog-to-Digital Converter, AN-191. Bernie Kute and Tom Harper, Oct
Microprocessor Training Schedule. Nov/Dec
Serial Data Transfers with SC/MP-II, AN-190. John Pacetti, Sep
The Data Bookshelf: Tools for the Design Engineer. Mar

DEPARTMENTS

Bit-Bucket. Jan 13, Feb 7, Mar 6, Apr 10, May 14, Jun 12, Jul/Aug 18, Sep 5, Oct 9, Nov/Dec 8
Consultants List. Feb 18, Jul/Aug 24
Editorials. Jan 13, Mar 7, Apr 13, May 12, Jul/Aug 18, Sep 5
Library. Jun 14, Sep 17
Library Submittal Form. Feb 16, Mar 16, Oct 13
Library Order Form. Feb 14, Mar 11, Jun 15, Sep 21

HARDWARE/SOFTWARE/LITERATURE UPDATES

Change to PACE Conversational Assembler. Sep 2
Correction to SC/MP Math Routines. Feb 6
Flip Your Floppy. Nov/Dec 12
Hardware Hint. Sep 3
How to Build a Digital Thermometer. Jun 5
Line-by-Line Resident Assembler for SC/MP MPU. Jun 1
Microprocessor Timesharing Software. Mar 15
NIBL Note. Sep 9
Notes from Australia. Ed Schoell, Oct 12
PACE Assembler for Data General Nova. Feb 15
PACE BASIC. Oct 5
PACE DOS Upgrade or Retrofit. May 17
Programming Miscellani/Programming Hints. Feb 6
Programming Tidbit. Apr 2, Jun 2, Sep 4, Nov/Dec 20
SC/MP Debugging Aid. J. B. Ross, May 19
SC/MP-II Retrofit Kit. May 19
SC/MP-II Retrofit Kit Firmware Changes. May 19
SM/PL – A High Level Language for IMP-16. Jun 2
Software Update. Nov/Dec 13
User Library Updates. Jan 5
Using Genlder with Debug. Mar 12

LITERATURE

Articles About SC/MP. Nov/Dec 20
Assembly Level Programming for Small Computers. Jan 12
Bug Books. May 15
Computer-Hobbyist Conference Book. Nov/Dec 10
Data Book on MOS/LSI. Oct 4
Information Systems Architecture. Nov/Dec 10
National's Free Literature Index. Nov/Dec 13
National's Microprocessor Literature. Mar 8
New CB Book. Jun 5
New CMOS Data Book. Nov/Dec 10
New Reference Directory for Personal and Home Computing. May 15
Recent Literature. Jan 12
Revised Data Book. Oct 4
Theory of Data Bases. Nov/Dec 10
Underground Buying Guide Tells Electronic Hobbyists Where to get it. Jun 14

MISCELLANEOUS

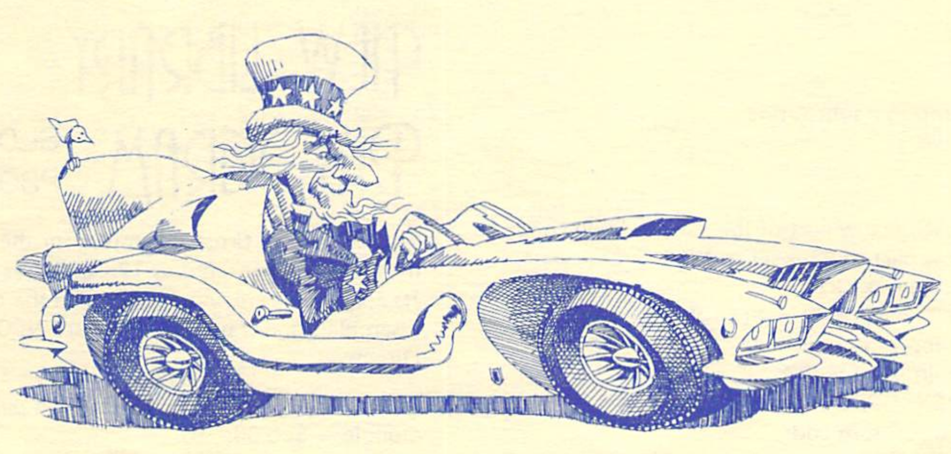
ACM Plans Hobbyist Group. Oct 5
Classifieds. Jan 3
Local Meeting. Mar 14
Microprocessor Consultants. Feb 18
Microprocessor Service Center. Mar 8
SC/MP Keyboard Kit Repair Policy. Jun 13
Services. Mar 10
1975 BIT-BUCKET Index. Jan 12
1976 BIT-BUCKET/COMPUTE Index. May 10

PROGRAM LISTINGS

IMP-16 SLØØ31 DORG. Jan 7
IMP-16 SLØØ38A TAPE. Apr 14
IMP-16 SLØØ4Ø 8Ø8Ø Cross Assembler. Jan 8
IMP-16 SLØØ5ØA PRTMOD. Jul/Aug 11
IMP-16 SM/PL TIMER PROGRAM. Jun 2
PACE BASIC – SLOT. Sep 4
PACE BASIC SLØØ48A BIORHYTHM. Jul/Aug 9
PACE SLØØ28A SQRT. Jan 6
PACE SLØØ32A DIVIDE. Feb 18
PACE SLØØ33A DELSEM. Feb 15
PACE SLØØ35A PRNTLM. Mar 9
PACE SLØØ36A BASCII. Mar 10
PACE SL0037A JITTER. Mar 9
SC/MP LCDS Punch Complemented Binary PROM Tape. Jan 6
SC/MP SLØØ41 SCSQRT. Apr 14
SC/MP SLØØ47A PLOT. Jun 3
SC/MP SLØØ49A REFLEX. Jul/Aug 11
SC/MP SLØØ52A Micro-Assembler. Sep 1
SC/MP SLØØ53A Loader for 8Ø8Ø. Sep 1

PRODUCT NEWS

A Family of RAM Support Circuits. Apr 10
A Number-crunching Microprocessor. Apr 13
A Powerful Development System for the 8Ø8Ø and SC/MP. Jan 1
COPS 4-Bit Microprocessor Family. Sep 15
Dictionary of SC/MP-II Support. Jul/Aug 16
European Product Announcement – ASM88. Sep 3
FFT Package for IMP-16P. Jan 1
Hand-Driven Paper Tape Reader. Nov/Dec 23
IMP-16 Microcomputer Gives 16 Megabit/sec DMA Transfer. Sep 3, Oct 5
IMP-16 16-Bit Microprocessor Family. Sep 13
M + S Analyser Makes SC/MP Transparent. Apr 1
Memory Boards Enhance National's Board Level Computer. Oct 15
Microcomputer New Product Releases. Sep 16
RAM I/O Chip. Nov/Dec 13
Microcomputer Systems. Oct 12
National Develops 65K MAXI-ROM. Nov/Dec 1
National Introduces Plug-Compatible Series/8Ø Microcomputer Boards. Oct 15
National Now Source for 29Ø1A. Nov/Dec 1
National's Authorized Programming Equipment. Nov/Dec 7
Need N8Ø8Ø Memory? Oct 12
Number-processing Microcontroller Eliminates Software Development Cost. Oct 8
PACE Disk Operating System. May 17
PACE DOS is Here. Sep 20
PACE Microprocessor Low Cost Development System. May 16
PACE, SC/MP Motherboard. Sep 3
PACE 16-Bit Microprocessor Family. Jul/Aug 19
Pocket TTY. Feb 15
Precision Programmable Voltage Regulators. Apr 2
Product Announcement – SYSCOM. Oct 8
SC/MP-II: A Fast, N-Channel, 8-Bit MPU. Apr 1
SC/MP-II is Here. May 9
SC/MP-II LCDS. Nov/Dec 7
SC/MP LCDS Users. May 9
SC/MP 8-Bit Microprocessor Family. Jul/Aug 20
Second Source for SC/MP. Sep 1
Second Source 8-Bit D/A Converts. Mar 5
Series/8Ø Microcomputer Family. Sep 15
Single-chip Character Generator. Apr 14
Super-Fast 4K RAMs. Mar 15
Support Circuits, Faster 8Ø8ØA's Added. Jun 1
Support Devices and Literature. Jul/Aug 21
7-Segment to BCD on a Chip. Mar 5
8 Bit-Wide TRI-SAFE PROMs Increase Memory Density. Nov/Dec 24
8-K EPROMs. Nov/Dec 7
265Ø 8-Bit Microprocessor Family. Sep 13
29ØØ Bit-Slice Microprocessor Family. Sep 14
4ØØ4 4-Bit Microprocessor Family (FIPS). Sep 16
8Ø8ØA 8-Bit Microprocessor Family. Sep 14
μPro-8Ø Supports INS8Ø8ØA MPUUs. Mar 15



COMPUTE EDITORIAL

We receive many questions about many things and in this column we will try to answer the most frequently asked questions.

First, we are asking everyone on our mailing list to return the requalification/update form from Volume 3, Number 10 issue of COMPUTE. We are doing this to update our mailing and find out more about your interests. And, by just filling out the form you will receive a SC/MP and/or an 8080 Quick Reference Guide. All members (even Lifetime) should fill out ths form and send it to the closest COMPUTE office.

Second, NIBL is a very popular program and we have received many questions regarding its availability. The following are the ways NIBL can be obtained.

1. Order SL0043 from the COMPUTE library. The cost is $15.00. This includes the object paper tape, listing and manual. The manual can be ordered separately for $5.00.

2. The parts below can be ordered from your local distributor.

a.	ISP-8F/351	NIBL ROMS (8-5214)	$75.00
b.	ISP-8F/352	NIBL ROMS (2-2316A)	$70.00

The 5214's can be put in the 4Kx8 SC/MP Application ROM/PROM card (ISP-8C-004B, $125). For information on using the 2-2316's with the SC/MP LCDS see COMPUTE, Volume 4, Number 1.

Thirdly, we have been receiving requests for SUPAK listings, which are not available. SUPAK was not written by National Semiconductor and can only be purchased as a ROM set. SUPAK is the line-by-line assembler, PROM tape generator for the SC/MP LCDS (order no. ISP-8F/111 $150.00).

The fourth item that deserves some attention is the JS pseudo instruction used with the SC/MP cross-assemblers. This instruction will generate the 7 bytes needed for a SC/MP subroutine jump. For example if we use P3 as the pointer and the subroutine is at address X'C005, then we could write either of the code segments below.

Using JS instruction

```
P3=3
SUB=X'COO5
JS  P3, SUB
```

Using 7 bytes to jump to a subroutine

```
P3=3
SUB=X'COO5
LDI L(SUB-1)  ; (AC) ← low part of the address minus one
XPAL 3        ; low part of P3 is set to X'04
LDI H(SUB-1)  ; (AC) ← X'CO
XPAH 3        ; high part of P3 is set to X'CO
XPPC 3        ; jump to the routine
                pointed at by P3
                PC ← COO5
                P3 ← return addr.
```

As you can see, the JS instruction functions like a macro call.

Finally, we have some requests. First, does anyone have any floating-point routines for SC/MP (INS 8060)? If so, please let us know. Second, when submitting articles or programs please specify the name and address of the person the complimentary COMPUTE membership should be sent to. Third, COMPUTE cannot accept purchase orders for less than $50.00 and will return them with a note requesting a check. This can delay processing your orders.

And last, our new phone number is (408) 737-6181.

Special Offer

SC/MP Kits (ISP8K/200) can be purchased for $50.00 each. These kits are available until our supply is exhausted. Send your order and check to Compute/208
National Semiconductor
2900 Semiconductor Drive
Santa Clara, CA 95051

Name ____________________
Address ____________________
City ______________ State ________ Zip ________

Quantity ordered: __________ SC/MP kits @ $50.00 each
__________ Total enclosed

AMD ALTERNATE SOURCE AGREEMENT

National has just signed a cross-licensing agreement with Advanced Micro Devices. The two companies will exchange mask sets and mask generation software for a series of standard low power Shottky devices.

National will give AMD its proprietary line of 20 pin octal buffers with adjacent input/output pins. Designated DM81LS95, 6, 7, 8, the devices are important interfaces with 8 bit microprocessors. National's DP8304B, a bidirectional transceiver, is also included in the exchange.

In return, National will receive from AMD the masks for the 74LS240, 1, 2, 3, and 4. These octal buffer/drivers with cross packaged input/output pins were originally manufactured by Texas Instruments.

Both companies plan product availability for July or August.

NEW LIBRARY PROGRAM

SL0054A
P8012ASM

This is an 8080 Cross-Assembler for the Universal Development System. It requires 12K and uses the disc routines. Its commands are compatible with the other NSC/UDS assemblers and it uses the standard NSC Assembler Directives.

Order no. SL0054 Source listing and paper tape object module — $25.00.

the Bit • Bucket

Dear Editors,

You have published my article on RAM loaders for small SC/MP systems in the Sept. 1977 issue (Vol 3, No 8) of "COMPUTE".

Since the original submission of the article, I have been able to make various improvements to our system. The Fast Loader is now shorter and faster; there is a new routine for inserting fresh bytes into RAM while moving existing code up in memory to make room for them, and one for starting execution of any program in RAM without using KITBUG or a terminal.

This enhanced system is described in a paper which has been accepted by "MICROPROCESSORS" (IPC, Guildford, UK), with a full listing of the whole code in the format we actually used with these loaders. It is expected that this will appear in the October issue.

One or both of these loaders may be suitable for Andy Nicoll, who was asking in a letter in your September 1977 issue for a SC/MP kit paper tape read program.

Yours Faithfully,
J. R. Stockton (Dr),
National Physical Laboratory,
Teddington TW11 OLW, UK.

Loader for KITBUG teletype.

```
Set PC to G01, G02, or G03.
If G02 or G03, set P1 to Start of Load Area (Default 0400).
If G03, set P2 as Stack Pointer (Default OFF7).

G01:  P1:=0400 – LDI 04 C4/04/ , XPAH P1 35/ , LDI 00 C4/00/ , XPAL P1 31/ ,
G02:  P2:=OFF7 – LDI OF C4/0F/ , XPAH P2 36/ , LDI F7 C4/F7/ , XPAL P2 32/ ,
G03:  P3->GHEX – LDI 00 C4/00/, XPAH P3 37/ , LDI DF C4/DF/ , XPAL P3 33/ ,
LAB:  Call GHEX then restore P2 – XPPC P3 3F/ , LD @+2(P2) C6/02/ ,
   Unless SLASH, try again – LDE 40/ , XRI SLASH E4/2F/ , JNZ LAB 9C/F8/ ,
   Move byte from GHEX to load area – LD -1(P2) C2/FF/ , ST @+1(P1) CD/01/ ,
   Continue for next byte – JMP LAB 90/F2) .

(hex 020 bytes)
This loader does not recognize QUERY and is stopped manually.
```

Dear COMPUTE,

Here's a suggestion for those SC/MP users who find themselves getting into endless loops in their Kit/Keyboard programs, but find it impractical and/or frustrating to set breakpoints in all their loops while debugging.

I've had great success using Sense Line "A", with interrupt enabled, as an external "debug" command. When things get lost and I don't reach a checkpoint "XPPC 3", I simply tap my switch and, voila, "- - - - - -". Then I can look at all the registers and find out what is going wrong. Of course, in order to be able to do this I have to remember to set the "interrupt enable" bit in location "OFFF" before I start up again.

Dave MacLEAN
985 Brussels St., Apt 18
Halifax, Nova Scotia
CANADA B3H 2S9

Dear COMPUTE:

We have developed an intermediate language for the SC/MP (INS8060) which we feel may be of interest to your readers. The language is of our own design and we call it "IL". The main design goal was to provide an easy programming language for data processing and to improve memory utilization without sacrificing machine speed or accessibility. IL is a table-driven language with the assembler generating the IL instructions as executable table elements implemented by MACROS or FORM statements. Since IL is generated via the assembler, IL statements may be interspersed with SC/MP assembly code if desired. The IL interpreter is coresident with the user program and interprets IL code generated by MACROS.

Because it executes assembly language type of instructions instead of having to parse high level language statements as in the case of NIBL, there is only a small sacrifice made in execution time due to the interpreters overhead.

The language itself is multi-stack oriented (data stack, subroutine return address stack, and register save stack). The data types include 1 byte, 2 byte (address), 3 byte (binary numeric 0-16,777,216), variable length string, and multidimensional tables of data records (files). All arithmetic is performed on 3 byte binary numbers. Included are add, subtract, multiply, divide, and MOD.

ASCII to INTEGER and INTEGER to ASCII are included as well.

A generous supply of stack manipulation instructions are included for manipulation of each data type (i.e., push string, concatinate, copy, allocate storage, etc.).

Subroutine calls save registers and may be N-levels deep. Conditional jumps using 16 bit addresses have also been implemented.

Perhaps the best feature of IL is that it is easily expandable by the user to include new IL instructions or user I/O drivers.

As a new user-statement is coded, the user may use the IL language to interpret his new IL statement. As an example, we used the IL multiply and add statements to generate the IL ASCII TO INTEGER decoding routines.

We have added many user statements and I/O drivers for our project and we have been exceptionally pleased with IL's performance. One test we performed showed the SC/MP performed several hundred 3 byte multiplies per second.

IL uses between 1200 to 2k bytes depending upon table and function requirements.

When coding data processing applications which mainly require computation, string, or table handling we estimate a 40% to 65% reduction of memory requirements and coding time. When coding I/O controllers and other machine code oriented routines we have found about a 10% to 50% reduction of memory and coding time.

We feel that IL has saved us man-months on our project and that it would be similarly useful to others. We are offering this program on a single payment license at $300.00 including object paper tape for the LCDS, a well documented source listing, a usage and modification guide, and a 30 day limited warrantee.

For an additional $50.00 we will include a source on paper tape. Additional information is available on request.

Cordially,

Eric Jameson
Software Services
B M G Corp.
811 So. 500 West
Bountiful, Utah 84010

Dear Georgia Marszalek,

Since my letter about a device which played chess on a T.V. receiver and stored chess games on cassette tape was printed in the issue of Compute magazine last February, a device based on the SC/MP II chip and other N. S. parts has been completed.

To the chess enthusiast it represents a facility for game analysis he has never had before: games may be recalled instantly and joined at any point, alternative lines of play considered and games may be annotated with verbal comments whilst they are actually being played. The reaction of those chess players known to me is strong enough to leave no doubt that there is a large and ready-made market for the device.

But I am not a manufacturer. I am an amateur and apparently a voice crying in the wilderness. A patent has been applied for and some letters written but without avail. Time moves so quickly in the electronics business and techniques are obsolete so quickly: I am afraid that a cheaper MPU with some integral ROM will make it necessary to design a Mark II version before anyone sees the original.

Sincerely,
Barry Savage,
17, Greenfields
PIDDLEHINTON,
Dorset (PUDDLETOWN 312)

Anyone wishing to build this system contact Barry at the address above.

Dear Georgia:

Thank you for your quick response to my letter. As you suggested I've inclosed a new check for Volumes one and two of COMPUTE.

May I suggest that you remind your readers that they must deal directly with their local distributors when purchasing hardware.

Again thank you for your help.

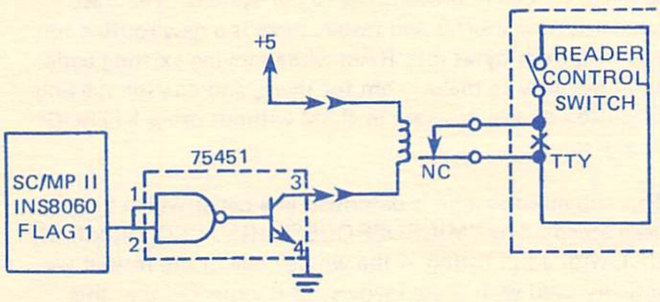
Most sincerely,
Stephen Taylor
BME Industries
1202 E. 17th – Suite 26
Santa Ana, CA 92701

Dear Georgia:

I recently built a test system from scratch using a SC/MP II and NIBL residing in 2 ea ROM chips and operating with 2k of 2102 RAM.

I am really impressed with the ease of programming with NIBL.

One thing the NIBL Manual did not mention was how to get your tapes to read in on the teletype. After a call to your NIBL experts, it was explained that they use Flag 1 for reader control. The figure shows how I implemented it.



By using opto isolators in the TTY interface I was able to get by with using a single supply for the whole system.

Another feature that I added was OR'ing a CRT terminal with the TTY. This way I can read in my programs, then switch off the TTY and continue programming on the CRT Terminal minus the noise.

I am looking forward to a reasonable priced assembler in ROM for the SCMP II.

Thank you.

Sincerely,
Philip M. Dooley
National Radio Astronomy Observatory
P.O. Box O
Socorro, New Mexico 87801

Dear COMPUTE:

Could you kindly refer to where I could get a circuit for an 80 character/line TV Monitor display? Or kit including character generator ASCII/EBCDIC?

Yours truly,

F. MacDonald
P.O. Box 50413
Dallas, TX 75250

Ed – Anybody able to help?

space is the place

if we don't make it off the planet
before we use all the energy available we'll never make it. Like old Pete, let's *prime the pump.*

FIRMWARE ROM BOOTSTRAP

by Robert K. Underwood
Reprinted with permission from the Homebrew Computer Club, P.O. Box 626, Mountain View, CA 94042.

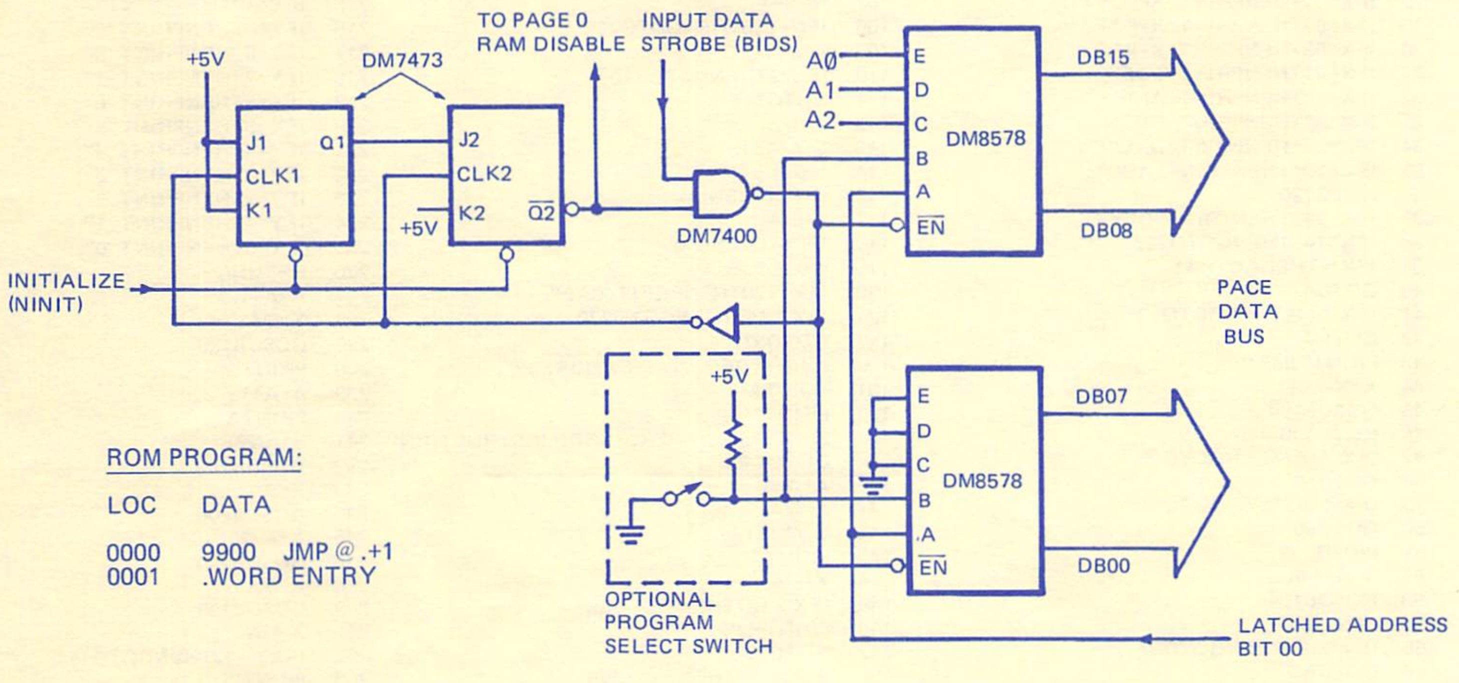
The National Semiconductor PACE microprocessor, like several others, restarts at location 0000. Therefore, at least that instruction must either be ROM, or must be manually loaded from a hardware front panel before the restart. Since my plans do not include a front panel, but I do need RAM in àt least part of base page memory space (0000-00FF), I was considering a full decode of locations 0000 and 0001 to bring in a 2-word ROM which would execute a jump to my main monitor program in high ROM.

A better solution is shown in the sketch: NINIT (negative initialization) loads a simple counter and opens a gate to allow the input data strobe (IDS) to gate the DM8578 bipolar ROMs onto the data bus. These pulses also clock the counter which, in my case, is arranged to allow exactly two pulses through before reaching its stable state. Base page RAM is disabled by the same signal that opens the gate to the ROMs by using this signal as one of the inputs to the high-order address comparator. When the counter reaches its stable state, the ROMs are permanently disabled (I could easily add a transistor switch to kill power) and the RAM is enabled until the next restart. This allows full unrestricted access to the base page RAM.

Location 0000 of the ROM program is a jump indirect through location 0001, which contains hex 7800, the entry point of my monitor program. Note the DM8578s are actually 32 words by 8-bits, so I have used only 1/16 of their capability. The other words could be used for jumps to other locations selected by a switch on the higher bits of the DM8578s. This would be useful to access a ROM based BASIC interpreter without transferring control through the monitor (drudgery).

My monitor program allows all of the usual control panel functions and several features not usually found on minimal systems. It is presently configured on a 2K X 16 ROM card with space for eight MM5204 ROMs. One pair of ROMs contains the I/O driver routines tailored to my hardware, but with entry and exit conditions identical with the standard routines in development systems. This allows me to run standard software intended for that use such as editor, assemblers, relocating loader etc.

While I have implemented this circuit only on the PACE, the general idea should be applicable to other processors. In the case of the 8080 and other 8-bit machines, only one DM8578 ROM would be necessary, though it might take 3 words instead of two to execute the jump. This would require a bit of work on the counter, but should take no more hardware.



ROM PROGRAM:

```
LOC     DATA

0000    9900  JMP @.+1
0001    .WORD ENTRY
```

SC/MP DISASSEMBLER

The SC/MP disassembler is written in NIBL, National's industrial basic language, and requires RAM at X 1000- 1FFF. The code to be disassembled can be loaded into another RAM card by the SC/MP monitor or can be read from a PROM stuck in an empty ROM socket on the memory board. The operator runs the program and types the start address for the disassembler to begin.

The disassembler uses the standard instruction codes. One difference, however, is that all jumps also have the decimal equivalent of the jump printed in <>.

Jim Derosa
NSC, FAE Boston

```
1   PRINT"SC/MP DISASSEMBLER"
2   PRINT""
3   GOTO130
5   IFX=0THENPRINT"HALT";
6   IFX=1THENPRINT"XAE";
7   IFX=2THENPRINT"CCL";
8   IFX=3THENPRINT"SCL";
9   IFX=4THENPRINT"DINT";
10  IFX=5THENPRINT"IEN";
11  IFX=6THENPRINT"CSA";
12  IFX=7THENPRINT"CAS";
13  IFX=8THENPRINT"NOP";
14  IFX=25THENPRINT"SIO";
15  IFX=28THENPRINT"SR";
16  IFX=29THENPRINT"SRL";
17  IFX=30THENPRINT"RR";
18  IFX=31THENPRINT"RRL";
19  IFX=48THENPRINT"XPAL PC";
20  IFX=49THENPRINT"XPAL P1";
21  IFX=50THENPRINT"XPAL P2";
22  IFX=51THENPRINT"XPAL P3";
23  IFX=52THENPRINT"XPAH PC";
24  IFX=53THENPRINT"XPAH P1";
25  IFX=54THENPRINT"XPAH P2";
26  IFX=55THENPRINT"XPAH P3";
27  IFX=60THENPRINT"XPPC PC";
28  IFX=61THENPRINT"XPPC P1";
29  IFX=62THENPRINT"XPPC P2";
30  IFX=63THENPRINT"XPPC P3";
31  IFX=64THENPRINT"LDE";
32  IFX=80THENPRINT"ANE";
33  IFX=88THENPRINT"ORE";
34  IFX=96THENPRINT"DAE";
35  IFX=104THENPRINT"ADE";
36  GOTO120
37  IFX=143THENPRINT"DLY";
38  IFX=143THENGOTO227
39  IFX>THENGOTO41
40  GOTO47
41  IFX<148THENGOTO43
42  GOTO47
43  PRINT"JMP ";
44  X=X-144
45  GOSUB194
46  GOTO140
47  IFX>147THENGOTO49
48  GOTO55
49  IFX<152THENGOTO51
50  GOTO55
51  PRINT"JP ";
52  X=X-148
53  GOSUB194
54  GOTO140
55  IFX>151THENGOTO57
56  GOTO63
57  IFX<156THENGOTO59
58  GOTO63
59  PRINT"JZ ";
60  X=X-148
61  GOSUB194
62  GOTO140
63  IFX>155THENGOTO65
64  GOTO71
65  IFX<159THENGOTO67
66  GOTO71
67  PRINT"JNZ ";
68  X=X-156
69  GOSUB194
70  GOTO140
71  IF X>167THENGOTO73
72  GOTO79
73  IFX<172THENGOTO75
74  GOTO79
75  PRINT"ILD ";
76  X=X-168
77  GOSUB194
78  GOTO227
79  IFX>183THENGOTO81
80  GOTO87
81  IFX<188THENGOTO83
82  GOTO87
83  PRINT"DLD ";
84  X=X-184
85  GOSUB194
86  GOTO227
87  IFX=196THENPRINT"LDI";
88  IFX=196THENGOTO227
89  IFX=212THENPRINT"ANI";
90  IFX=212THENGOTO227
91  IFX=220THENPRINT"ORI";
92  IFX=220THENGOTO227
93  IFX=228THENPRINT"XRI";
94  IFX=228THENGOTO227
95  IFX=236THENPRINT"DAI";
96  IFX=236THENGOTO227
97  IFX=244THENPRINT"ADI";
98  IFX=244THENGOTO227
99  IFX=252THENPRINT"CAI";
100 IFX=252THENGOTO227
101 N=192
102 Z=1
103 C=1
104 I=0
105 P=0
106 IFX=NTHENGOTO160
107 N=N+1
108 IFN=256THENGOTO239
109 Z=Z+1
110 IFZ=9THENGOTO112
111 GOTO116
112 Z=1
113 C=C+1
114 I=0
116 IFZ=4THENI=1
117 P=P+1
118 IFP=4THENP=0
119 GOTO106
120 IFX=120THENPRINT"CAE";
121 IFX<143THENGOTO230
122 GOTO37
130 PRINT"DEC. START ADDRESS";
131 INPUTA
132 PRINT""
133 PRINT"DEC.  HEX.  CODE INSTRUCTION"
134 GOTO240
140 A=A+1
141 X=@A
142 GOSUB199
143 PRINT"    <";
144 X=@A
145 IFX>127THENX=X-256
146 PRINTX,">";
147 GOTO230
150 IFX>127THENX=X-256
151 PRINTX;
152 RETURN
154 X=@(A+1)
155 GOSUB200
156 PRINT"  ";
157 X=@A
158 GOTO37
160 IFC=1THENPRINT"LD ";
161 IFC=2THENPRINT"ST ";
163 IFC=3THENPRINT"AND ";
164 IFC=4THENPRINT"OR ";
165 IFC=5THENPRINT"XOR ";
166 IFC=6THENPRINT"DAD ";
167 IFC=7THENPRINT"ADD ";
168 IFC=8THENPRINT"CAD ";
169 IFI=1THENPRINT"@";
170 A=A+1
171 X=@A
172 GOSUB150
173 PRINT" (";
174 X=P
175 GOSUB194
176 PRINT")";
177 GOTO230
180 IFA<10THENPRINT" ";
181 IFA<100THENPRINT" ";
182 IFA<1000THENPRINT" ";
183 IFA<10000THENPRINT" ";
184 RETURN
194 IFX=0THENPRINT"PC";
195 IFX=1THENPRINT"P1";
196 IFX=2THENPRINT"P2";
197 IFX=3THENPRINT"P3";
198 RETURN
199 PRINT"    ";
200 Y=15
201 IFX>=Y*16THENGOTO204
202 Y=Y-1
203 GOTO201
204 X=X-Y*16
205 GOSUB210
206 Y=15
207 IFX>=YTHENGOTO210
208 Y=Y-1
209 GOTO207
210 IFY=15THENPRINT"F";
211 IFY=14THENPRINT"E";
212 IFY=13THENPRINT"D";
213 IFY=12THENPRINT"C";
214 IFY=11THENPRINT"B";
215 IFY=10THENPRINT"A";
216 IFY=9THENPRINT"9";
217 IFY=8THENPRINT"8";
218 IFY=7THENPRINT"7";
219 IFY=6THENPRINT"6";
220 IFY=5THENPRINT"5";
221 IFY=4THENPRINT"4";
222 IFY=3THENPRINT"3";
223 IFY=2THENPRINT"2";
224 IFY=1THENPRINT"1";
225 IFY=0THENPRINT"0";
226 RETURN
227 A=A+1
228 X=@A
229 GOSUB199
230 PRINT""
239 A=A+1
240 PRINTA;
241 B=A/255
242 GOSUB180
243 X=B
244 GOSUB200
245 X=A-256*B
246 GOSUB200
248 X=@A
249 GOSUB199
251 X=@A
252 IFX>142THENGOTO154
253 PRINT"    ";
254 GOTO5
255 END
```

Call For Papers

Second Annual Rocky Mountain Symposium on Microcomputers: Systems, Software, Architecture.
August 27–August 30, 1978, Pingree Park, Colorado State University, Ft. Collins, Colorado.

Theme: "Significant Roles for Microcomputers in the Coming Decade."

You are invited to submit three copies of your paper to the program chairman:

Mike Tindall
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
(303) 491-7096, 491-5792

Original papers are solicited in the following areas:

Numerical Computation
Including: numerical algorithms, implementation for efficient, accurate computation, and interaction with microcomputer design.

Social Impact
Including: the role of software and its attendant legal aspects, very large networks for the home, industrial and commercial use, privacy, electronic mail, and funds transfer.

Microcomputer Software
Including: transportability of software, development of universal software, systems software, and applications software.

Microprocessor Architectures
Including: implications of increasing packing density, high level language oriented features, standardization.

Multiple Microprocessor Systems
Including: distributed system design principles, reconfigurable systems, interconnection structures, software, languages, and applications for multiple microprocessor systems.

Military Applications
Including: avionics, fire control, system management, and secure communication networks.

Deadline for Submission: May 1, 1978
Notification of acceptance will be given by June 15, 1978. Papers accepted for the conference will be published in the Symposium Proceedings and distributed at the conference.

For Further Information
Contact the Symposium Co-Chairmen:

Mike Andrews
Department of Electrical Engineering, (303) 491-5767
Steve McCormick
Department of Mathematics, (303) 491-5763
at Colorado State University
Fort Collins, CO 80523

Upcoming Conferences and Conventions

Compcon 78: February 28–March 2 at the Jack Tar Hotel in San Francisco. Sponsored by the Institute of Electrical & Electronic Engineers (IEEE).

The theme of this convention is "Computer Technology: Status, Limits, Alternatives." It will cover a variety of topics in hardware, software, and applications. An innovation this year is a program on personal computing, with admission at reduced rates.

For more information, contact: Jean Sherman, COMPCON 78 SPRING, IBM Corp. G32/006, 5600 Cottle Rd., San Jose, CA 95193.

Percomp '78: April 28–30 at the Long Beach Convention Center, Long Beach, CA. Exhibitors and speakers on a variety of personal-computing topics from home energy management and music generation to software design and marketing. Co-sponsored by the International Computer Society/SCCS and the Rockwell Hobbyist Computer Club.

Contact Percomp '78, 1833 E. 17th St., Suite 108, Santa Ana, CA 92701.

History of Programming Languages: June 1–3 in Los Angeles. Sponsored by SIGPLAN. Lectures and discussions led by key contributors to the development of languages like Fortran, Cobol, APL, and LISP. The keynote speaker will be Captain Grace Murray Hopper.

Contact Billy G. Claybrook, Department of Computer Science, Virginia Polytechnic Institute, Blacksburg, VA 24061.

NorthWest 78: June 15–16 at the University of Victoria, British Columbia. Sponsored by the Vancouver and Victoria Sections of the Canadian Information Processing Society (CIPS), and the Puget Sound Chapter of the ACM.

The conference has two themes: use of computers in regional industries (forestry, paper, mining, oil, fishing, etc), and use of computers in municipal, state or provincial, and federal governments.

Contact NorthWest 78, POB 570, Ganges, BC VOS 1E0.

ACM 78: Call for Papers. To be held December 4–6 in Washington, DC. Deadline for submission of paper or session proposals, July 1. Submit five copies to: Gerald L. Engel, Department of Mathematics and Computing Sciences, Old Dominion University, Norfolk, VA 23508.

There will be a special program this year on applications and policy issues relating to computers in Federal Government. Paper proposals for this program should be sent to: Dennis M. Conti, Systems & Software Division, National Bureau of Standards, Washington, DC 20234.

BACK ISSUES

Back issues of COMPUTE are available for $1.00 each. Please send your check or money order made payable to COMPUTE to:

National Semiconductor/208
2900 Semiconductor Dr.
Santa Clara, CA 95051

Mark on the form below, the copies you wish to receive.

Please circle the issues desired and return this form with your check.

1978 Vol 4

#1 #2

1977 Vol 3

#1 #2 #3 #4 #5 #7 #8 #9 #10

($8.00 for entire set, Vol. 3, #6 is out of print).

1976 Vol 2

#1 #2 #3 #4 #5 #6 #7 #8 #9 #10 #11 #12

($10.00 for entire set).

1975 Vol 1

#1 #2 #3 #4 #5 #6

($5.00 for entire set).

Total amount enclosed . . . . $ ________

Name ________________________________

Company ________________________________

Address ________________________________

____

____

Simplify CRT Terminal Design with the DP8350*

Application Note 108
Helge H. Mortensen

INTRODUCTION

This application note is a description of a "low cost" CRT data terminal card design, based upon National's CRT Controller (DP8350) and 8-bit N-channel microprocessor (SC/MP). The terminal has a minimum parts count and implements all TTY functions. Even with this minimum number of parts, the terminal provides some "smart" features by efficiently utilizing the available hardware. Screen scroll, RS-232C interface and adjustable baud-rate (up to 1200-baud) are featured on the card. Higher baud-rates are available on a word-by-word basis if the RS-232-handshake signal is used. The design also demonstrates use of 2 new microprocessor-interface parts: the Asynchronous Communications Element (ACE), for serial I/O; and the RAM Input/Output (RAM I/O), for keyboard scanning and scratch pad memory. A 2-kilobyte video RAM is implemented with four 1024 x 4-bit, static RAM chips (MM2114), and dot generation uses the DM8678 5 x 7 Character Generator.

The card is self-contained except for the CRT monitor and power supply. It holds a keyboard and monitor-interface circuitry. Monitors requiring separate video and sync signals (Ball Brothers), and those requiring composite video (Motorola) are accommodated.

System Architecture

Since system cost is typically somewhat proportional to parts count, arriving at a minimum parts count solution has been a goal throughout this design effort.

A full-blown CRT terminal is shown in *Figure 1* and its low cost counterpart in *Figure 2*. Address decoding details are omitted in both cases.

*DP8350 is cross referenced as an INS8276.

Removing overhead circuitry shown in *Figure 1*, and making use of the TRI-STATE® concept greatly facilitated the parts reduction effort.

Obviously, extreme time conflicts for communicating on the system busses are created because all essential parts require access. Let us investigate this problem a little further.

Because the CRT Controller does the CRT display refresh function, it must have access to a memory containing current data for display. This memory may be a shift register (octal, 80-bit line buffer in *Figure 1*) which is loaded at the first video line in a character row and then recirculated for the number of video lines in that character row. Using such a line buffer allows the microprocessor access to the system busses for more than 90% of the video time (screen time). On the other hand, by removing the buffer, the refresh circuit needs direct memory access (DMA) during video display time.

However, with the bidirectional data buffer and the TRI-STATE address buffer in the system, the situation is not yet too serious. (We are only preventing the SC/MP microprocessor from updating video RAM data or using the scratch pad during character display time.) Instruction fetch (ROM) and keyboard scanning are still not affected. However, with the saving of the data buffer and the address buffer, a well-organized "time share" of the busses is required. How does this limited bus access affect the time-critical features such as scroll and high baud rate?

Scroll

Several scrolling methods may be implemented with the CRT Controller. The most straightforward is a rewrite of memory. This requires long processing time and bus access and is not feasible with the minimum hardware indicated in *Figure 2*. Sensing when the CRT is scanning

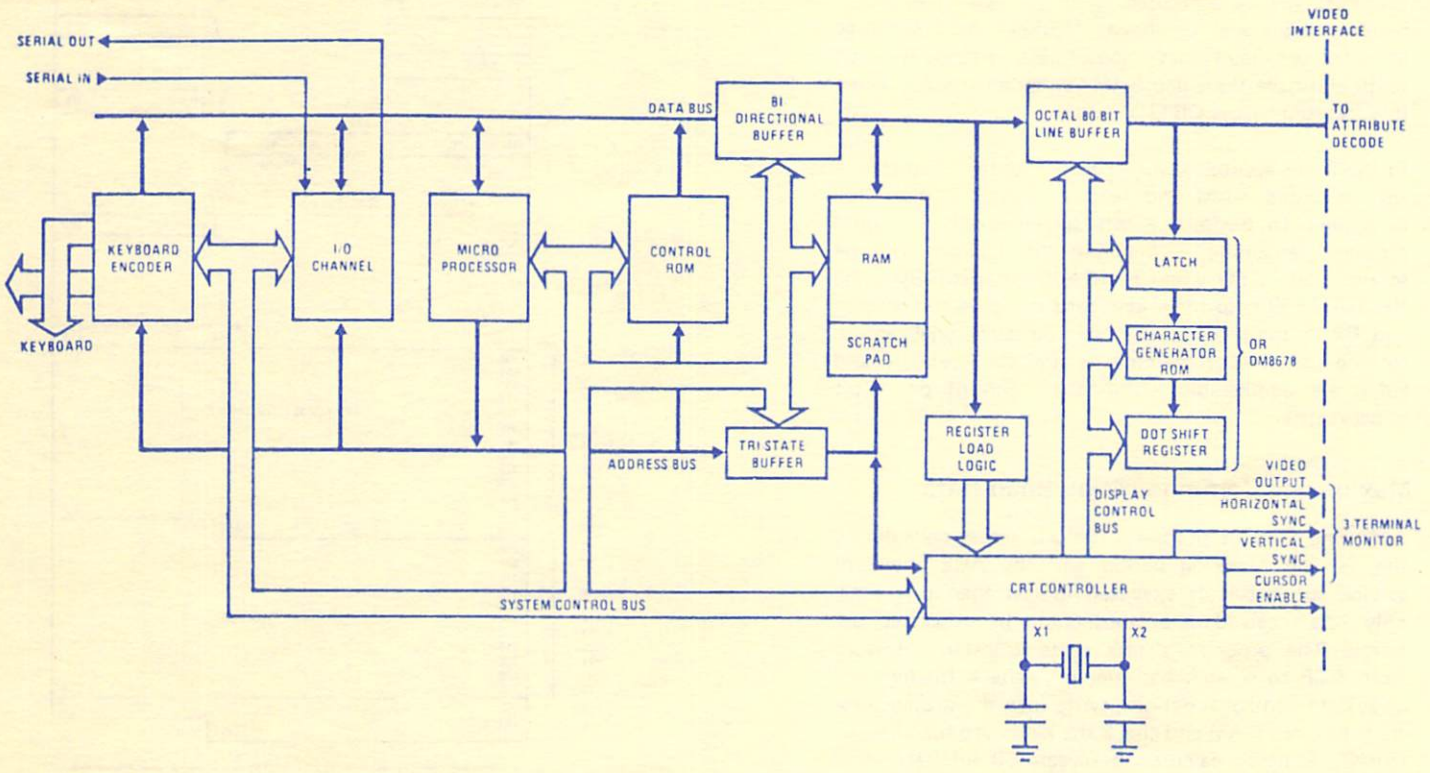


Figure 1. System Block Schematic Using Line Buffer, Address & Data Buffer

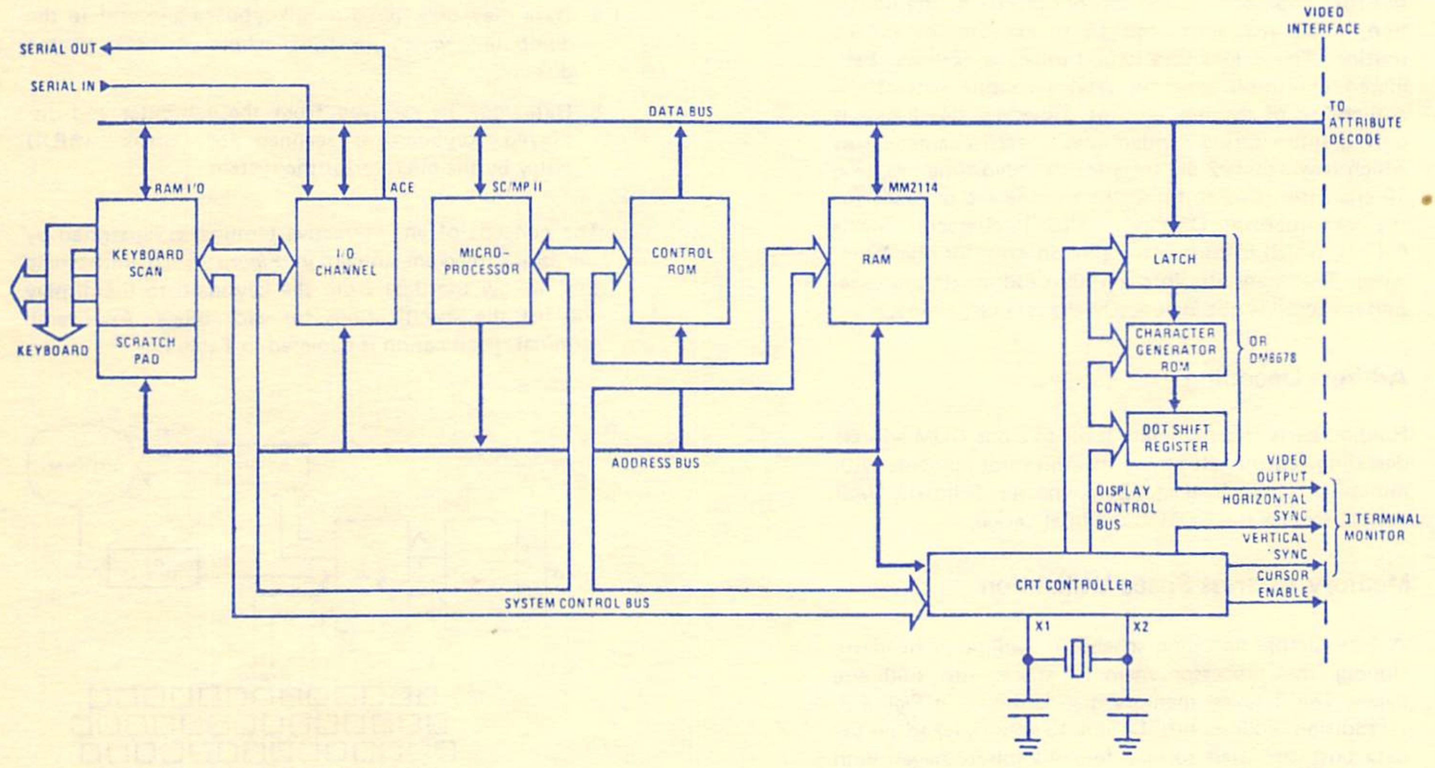


Figure 2. System Block Schematic for the Low Cost Terminal

character-row 24 and then loading a new "row start" requires additional overhead circuitry. An alternative approach is to load a new "top of page" address for each scroll and have the video RAM "wrapped-around" when it is accessed by the CRT!

In the latter approach, the processor only has to clear a row in video RAM and load a register in the CRT Controller to perform a total-screen scroll. The only problem remaining is handling the location of the scratch pad in the video RAM address space. By using the RAM I/O chip for a keyboard scanning and scratch pad RAM, the problem is solved. An additional feature for the software programmer is that the keyboard and RAM are addressable within the reach of one 8-bit index register.

Maximizing Communication Baud Rate

Assuming that the processor has bus access only during the vertical blanking period and the ACE interrupt service subroutine is executed in less than this time, only one received data word could be processed per frame! The processor's task is to transfer the word from ACE to scratch pad memory, check for terminal or system control functions, write into the proper locations in video RAM and check the keyboard for "Break" (BRK). A quick calculation reveals 60 bits/second as the maximum baud rate! To improve communication speed, the processor must have bus access during the video frame scan time. Three of the 10 scan lines making up a character row are blanked except during cursor time. Using these three lines (minimum decoding required) for processor bus access during video time allows us to communicate at 1200 baud.

Note that the baud rate is limited by the frame rate in the first approach; in the second approach, the limitation is the real time required to execute the service routine. The calculation is performed as follows. Estimated execution time for service routine with 100% availability of the bus is 2 ms. However, bus access is only granted during 3 video lines in each character row which is worth 192 μs. In terms of video time, we need 10 character rows to finish the routine and be ready for the next interrupt. Display time for 10 character rows is 6.4 ms, which in turn is the time interval for one 10-bit word. This translates into a 1600-baud maximum capability if scroll is not included in the service routine.

Address Decoding

Holding parts to a minimum leads to a one ROM address decoding scheme. However, this does not coincide with minimum cost. Instead, 2 low-power Schottky MSI devices replace the ROM in the final design.

Memory Address Space Utilization

A very simple decoding scheme is facilitated by partitioning the processor memory space into 500-byte pages. The detailed memory map is shown in *Figure 3*. In addition, address bits 12 and 13 (multiplexed on the data bus), are used to map four 4-kilobyte pages, with the first page dedicated to processor peripherals and the following 3 pages dedicated to register loading of the CRT Controller. The 3 CRT Controller registers to be loaded are "top of page", "row start" and "cursor".

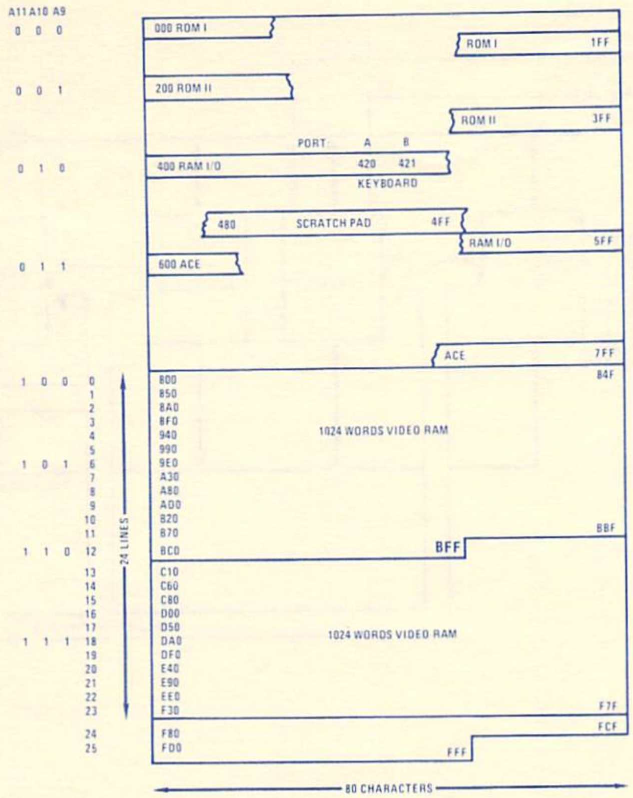


Figure 3. Memory Map

Discussion of Specifications

A device used to communicate with a computer is called interactive if it has the following properties:

a. Data may be entered on a keyboard and sent to the computer, which in turn echoes it back to the display.

b. Data may be received from the computer and displayed; keyboard is scanned for "Break" (BRK) entry by the operator of the system.

The concept of an interactive terminal is illustrated by the block diagram shown in *Figure 4*. To understand this, follow the data from the keyboard to the display and list the specifications for each block. An overall terminal specification is depicted in Table I.

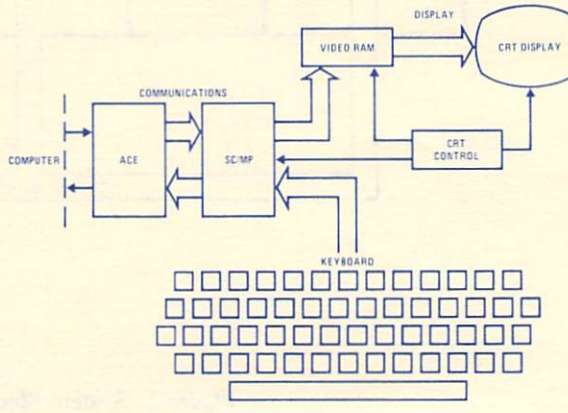


Figure 4. Block Diagram of an Interactive Terminal

TABLE 1. TERMINAL SPECIFICATIONS

Keyboard	
Style	Typewriter
Characters/code set	64/ASC II
Cursor controls	6
Keyboard encoder	Software
Communication	
Mode	Full duplex, half duplex option
Technique	Asynchronous
Communications protocol	ASC II
Code	ASC II
Bits/character	10/11
Speed, bits/second	110 to 1200 (19,200 word-by-word)
Operator selectable speeds	4
Format	Character
Terminal interface	RS-232, 20 mA current loop
Display	
Display positions, characters/display	1920
Display arrangement (line x characters)	24 x 80
Total display symbols	64
Symbol formation	5 x 7 dot matrix
Reverse video	Cursor and whole screen
Scrolling	Yes
Cursor type	Block, reverse video
Cursor position	Down, left, right, home and return, back space.

Keyboard

The keyboard is a copy of a standard teletypewriter with two-key rollover. The 54 keys can be broken into alphanumeric, punctuation, symbols, cursor control, and system control keys. The processor scans the keys at all times and translates any key closure into a unique code (ASC II), which is sent to the input/output channel for serial transmission to the computer. It should be noted that the RAM I/O chip has the capability of scanning 64 keys (8 x 8).

Communications

The input/output channel is based upon the Asynchronous Communication Element (ACE). This integrated circuit performs parallel-to-serial conversion of the data received from the keyboard, and serial-to-parallel conversion of data sent from the computer for display on the screen. When the system is initiated (power-up), the on-chip programmable baud generator is loaded with the desired baud rate (switch selectable). Start, stop, and parity bits are appended or deleted in this block of the system, depending on the direction of data flow. All control signals for the standard RS-232C interface are likewise generated here. Standard electrical specifications for RS-232C and 20 mA current loop are met by adding dedicated interface parts.

Display

After the data is received from the computer, it is stored in the video RAM. The CRT Controller chip refreshes the display at 60 Hz by sequentially addressing the video RAM; 1920 addresses are generated to fetch data for 24 lines of 80 characters. The standard 64-character ASC II set is displayed using a 5 x 7 dot-matrix block for each character. Data is entered from left to right and from top to bottom, until the screen is full. After that, upward scrolling with top-line overflow and newly cleared bottom line takes place automatically with line feed.

Software

A detailed flow chart of the software is shown in *Figure 5*. It is set up to service 3 major functions: a) initialize the system; b) scan the keyboard and c) service the ACE upon interrupt request.

a. Initialization
The video RAM is cleared and the cursor is loaded at the upper left corner of the screen. ACE is set up with the desired baud rate and the interrupt enable flag is armed.

b. Keyboard Scan
The keyboard is first checked for "Any Key Down" status. If positive, the keyboard is scanned and the binary code (ASC II) is computed by the program and read to ACE.

c. ACE Interrupt Service Routine
When its receiver buffer is full, the ACE puts out an interrupt request. The SC/MP immediately suspends keyboard scanning and reads the buffer register. The main portion of this routine is checking incoming data for control functions and updating the video RAM and the CRT Controller registers. It should be noted that the need for executing this routine is the limiting factor for high baud rate communications.

Hardware

The detailed hardware implementation is shown in *Figure 6*. The CRT Controller grants the SC/MP microprocessor bus access during blanked scanlines and vertical blanking interval by logically OR-ing line counter outputs with the vertical blanking pulse and using this signal as a bus-available signal. The CRT Controller is held off the bus by disabling the TRI-STATE address output. This is done by applying logical "0" to the RAM address enable pin of the CRT Controller, the SC/MP then takes the bus as needed.

Sense-A of SC/MP is used as an interrupt request input whenever received data is available in the receiver buffer register of the ACE. The interrupt service routine is executed during vertical blanking and "inactive" video time as indicated above.

The keyboard is sensed for "Any Key Down" (under program control) by reading Port-B of the RAM I/O chip. Upon a positive result, the keys are scanned by a special sequence for key identification and encoding.

Mechanical

A PC board layout and its assembly is shown in *Figure 7*. Note that the keyboard is mounted directly on the card.

Acknowledgment

The development of the terminal involved significant contribution by a large number of people. The author would particularly like to express his appreciation to Len Bryson for software development and Dana Knight for invaluable suggestions and ideas during the hardware design phase.

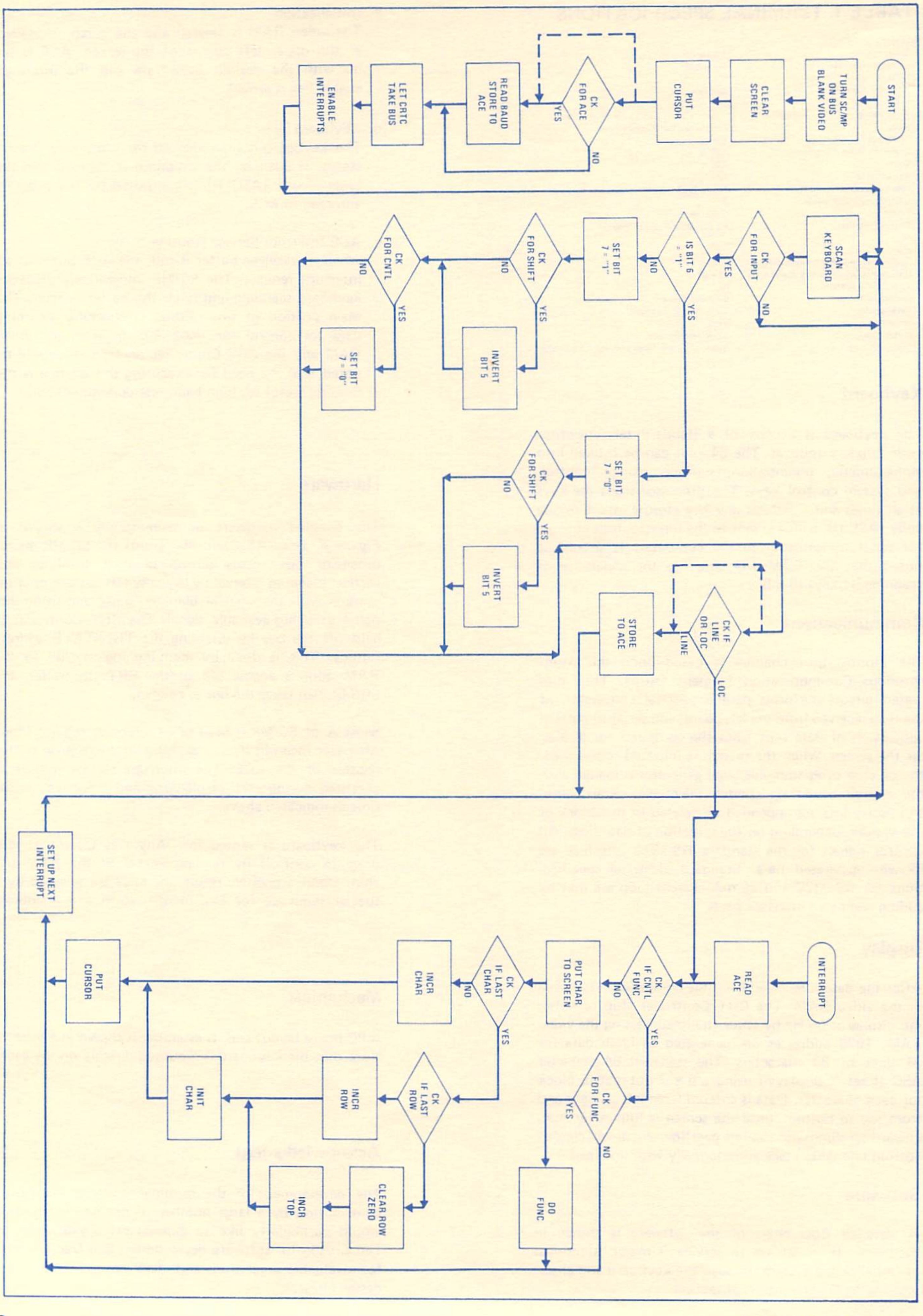


Figure 5. Low End Terminal Flow Chart

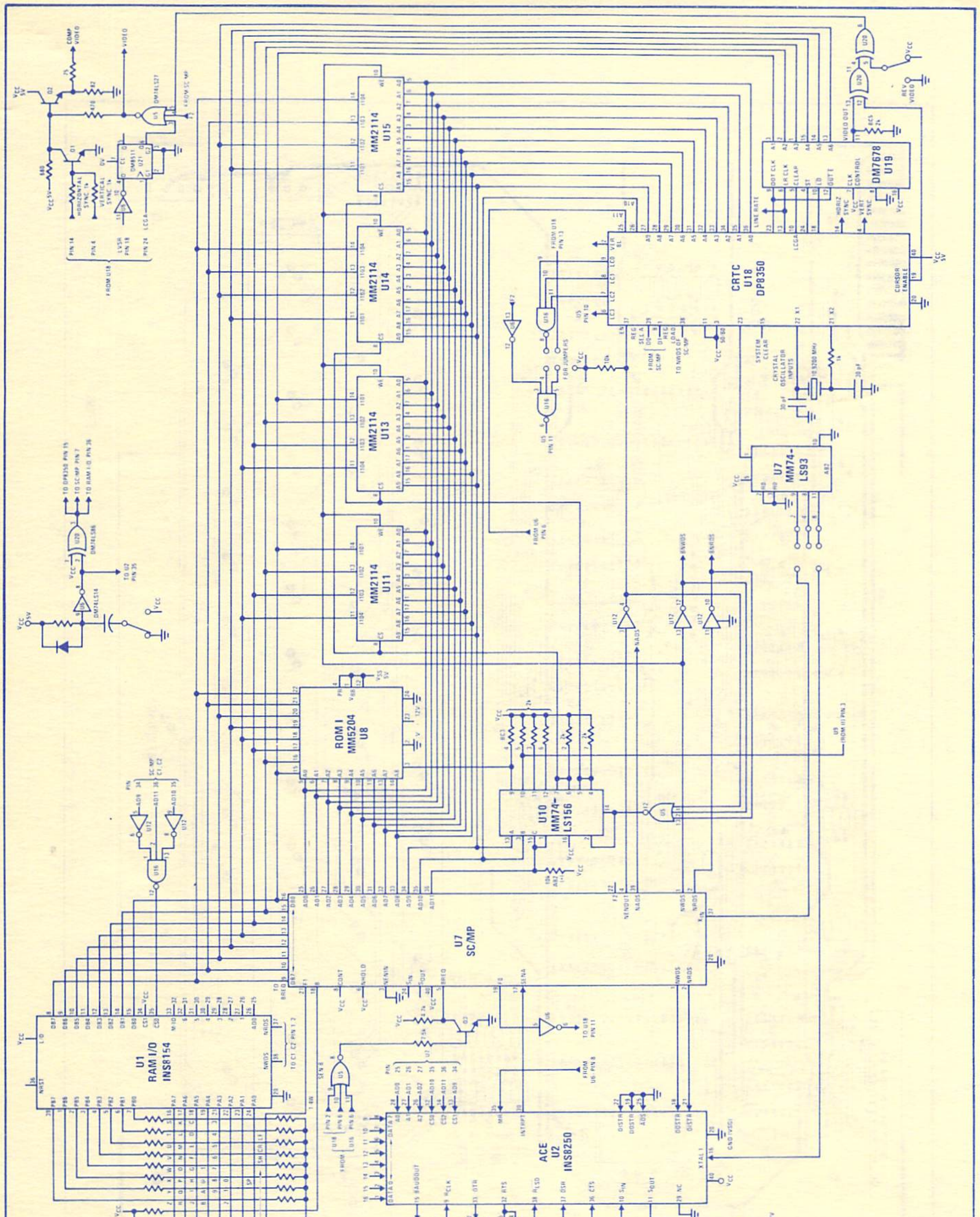


Figure 6. Low Cost CRT Terminal

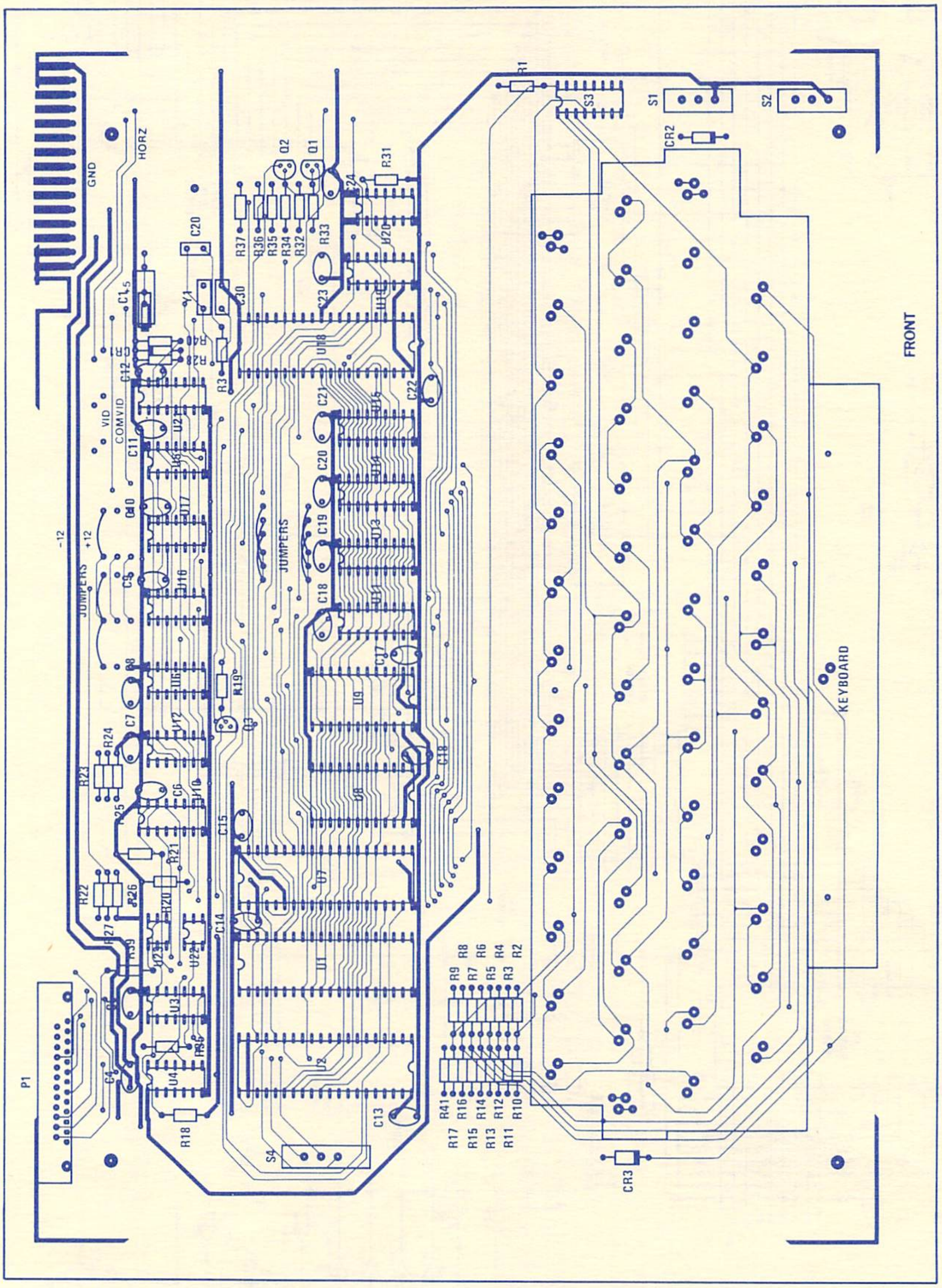


Figure 7A. PC Board Layout Assembly

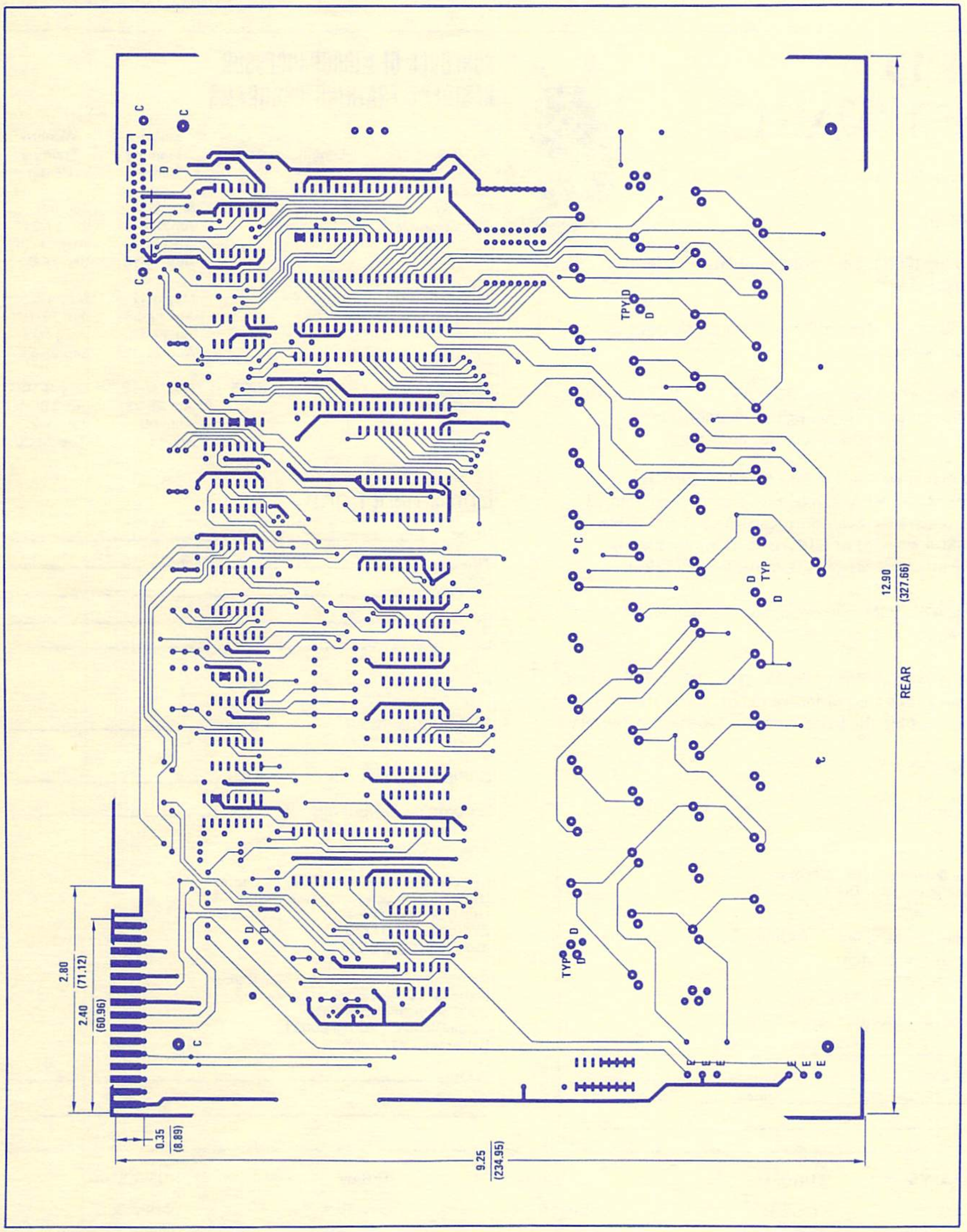


Figure 7B. PC Board Layout Fabrication

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 317
SUNNYVALE, CALIF.

HARD SOFTWARE

No, that's not a contradiction in terms.

It's a new ROM.

A ROM so big (65K) it can hold complete high level language interpreters.

Hardware that does the work of software. We call this concept Silicon Software.

Officially known as the MM5235 MAXI-ROM,™ the little devil will hold more than enough programmed memory (8K Bytes) for most microprocessor applications.

A perfect match for 8-bit microprocessors such as our 8080A, the MAXI-ROM can be ordered pre-programmed with LLL (Lawrence Livermore Lab) BASIC (INS8298, for the 8080A system) or NIBL BASIC interpreter for the SC/MP microprocessor system (order INS8295).

The 65K MAXI-ROM has completely static operation, and does not require a clock.

National offers a full line of ROMs, from 1K to 65K with a fast five-week turn-around (guaranteed) on low density units from the time the pattern code is received to delivery.

So get your pen out.

Don't forget now.

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, CA 95051

Gentlemen, please send me further information about your:

☐ new 65K MAXI-ROM
☐ entire ROM line
☐ 8298 LLL BASIC for 8080
☐ 8295 NIBL BASIC for SC/MP

Name ____________________
Address ____________________
City ____________ State ________ Zip ______

SCHEDULE OF MICROPROCESSOR RESIDENT TRAINING PROGRAMS

Course	Length	Cost	Eastern Training Center	Western Training Center
Microprocessor Fundamentals	4½ days	$475	May 1-5 Jun 5-9 Jul 10-14 Aug 14-18	Jun 5-9 Jul 17-21 Aug 21-25 Sep 18-22
8060 (SC/MP) Applications	4½ days	$475	May 8-12 Jun 12-16 Jul 17-21 Aug 21-25	May 1-5 Jun 12-16 Jul 24-28 Sep 25-29
Complex Peripherals	3 days	$395	May 22-25 Jun 26-28 Aug 7-9 Sep 11-13	May 8-10 Jun 19-21 Jul 31-2 Aug 28-30

Enrollment Form

Name ____________________
Title ____________ Telephone ____________
Company ____________________
Address ____________________
____

Course Selected

☐ Microprocessor Fundamentals Date ____________
☐ 8060 Applications Date ____________
☐ Complex Peripherals Date ____________

Training Center

Eastern Microprocessor Training Center
National Semiconductor Corporation
One DeAngelo Drive
Bedford, MA. 01730
Tel: (617) 275-8530

Western Microprocessor Training Center
National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 737-6453

☐ Other ____________________
____

UNITED STATES	EUROPE	AUSTRALIA	JAPAN	HONG KONG
COMPUTE/208	COMPUTE	COMPUTE	COMPUTE	COMPUTE
National Semiconductor Corp. 2900 Semiconductor Drive Santa Clara, California 95051 (408) 737-6181 TWX: 910-339-9240	National Semiconductor GmbH 808 Fuerstenfeldbruck Industriestrausse 10 West Germany Telephone: (08141) 1371 Telex: 05-27649	NS Electronics Pty. Ltd. Cnr. Stud Rd. & Mtn. Highway Bayswater, Victoria 3153 Australia Telephone: 03-729-6333 Telex: 32096	NS International Inc. Miyake Bldg. 6F, I-9 Yotsuya Shinjuku-Ku Tokyo 160, Japan Telephone: 03-355-3711 Telex: J28592	NS Electronics (HK) Ltd. 4 Hing Yip Street, 11th Floor Kwun Tong Kowloon, Hong Kong Telephone: 3-411241-8 Telex: 73866 NSE HK HX